THÈSE

PRÉSENTÉE À
L'ÉCOLE NATIONALE DES PONTS ET CHAUSSÉES

POUR OBTENIR LE TITRE DE
DOCTEUR DE L'ÉCOLE NATIONALE DES PONTS ET CHAUSSÉES

Spécialité : **INFORMATIQUE**

PAR

François WAHL

Sujet de la thèse :

UN ENVIRONNEMENT D'AIDE AUX INGÉNIEURS BASÉ SUR UNE ARCHITECTURE EN TÂCHES ET SUR UN MODULE DE VISUALISATION DE COURBES

APPLICATION À LA CONCEPTION DE PROCÉDÉS DE RAFFINAGE

Soutenue le 5 décembre 1994 devant le jury composé de :

MM.	*Michel GONDRAN*	*Directeur de thèse*
	François RECHENMANN	*Rapporteur*
	Marc AYEL	*Rapporteur*
Mme	*Brigitte TROUSSE*	*Examinateur*
MM.	*Jacques KOULOUMDJIAN*	*Examinateur*
	Bertrand BRAUNSCHWEIG	*Examinateur*

REMERCIEMENTS

M. **Alain Bamberger**, par son appui continu, a permis la réalisation de cette thèse. En tant que directeur de la Division Informatique et Mathématiques Appliqués de l'Institut Français du Pétrole (IFP), il m'a concédé le temps qu'il fallait pour mener à bien ce travail dans un cadre professionnel. Je pense que cette réflexion croisée entre les domaines d'application et la recherche est extrêmement profitable. Qu'il soit ici vivement remercié d'avoir offert ce cadre propice.

Je remercie les membres du jury d'avoir bien voulu juger cette thèse.

- M. **Michel Gondran**, directeur de thèse, directeur du département TIEM (Traitement de l'Information et Études Mathématiques) à EDF, a su dégager le temps nécessaire, malgré ses responsabilités variées. Son appui a été particulièrement précieux pendant la phase de rédaction.

- Mme **Brigitte Trousse**, chargé de recherche à l'INRIA, a encadré ce travail pour le compte de l'école des Ponts, et du laboratoire CERMICS. Elle a su faire le lien entre un monde de l'Intelligence Artificielle théorique et ses applications pratiques.

- M. **François Rechenmann**, directeur de recherche à l'INRIA Rhones-Alpes, et M. **Marc Ayel**, professeur d'informatique à l'université de Savoie, me font l'honneur d'être rapporteurs de cette thèse.

- M. **Jacques Kouloumdjian**, professeur d'informatique à l'INSA de Lyon, m'a assisté pendant le temps du DEA. Je garde un souvenir reconnaissant de son aide.

- M. **Bertrand Braunschweig**, responsable du groupe de compétence Intelligence Artificielle de l'IFP, a constamment soutenu ce travail. Ses qualités rédactionnelles constituent toujours une assistance précieuse.

Une thèse est aussi par bien des aspects un travail collectif. La partie application doit beaucoup à mes contacts avec les experts de l'IFP, et parmi eux je tiens à distinguer MM. **Alain Billon** et **Pierre-Henri Bigeard**, responsable du département des unités pilotes.

La partie 'interface' de l'application PRINCE qui sous-tend ce travail, doit beaucoup à un stagiaire du DESS IA de Chambéry, M. **Luc Leroy**.

D'autres très nombreuses contributions mériteraient d'être signalées. Merci à tous.

À Anaïs, Colin, Claire-Ariane et Christiane

RÉSUMÉ

Dans le domaine du génie chimique, les ingénieurs tracent des courbes pour analyser les données recueillies. Une fois validée, cette connaissance est exploitée, en combinaison avec d'autres savoirs, sous forme de tâches. Cette thèse présente une architecture capable d'enchaîner n'importe quel type de tâches et de visualiser des courbes, appliquée à un problème d'aide à la conception de procédé de raffinage.

L'architecture proposée repose sur une analyse objets des raisonnements, où figurent les notions de relations (inversibles ou non) et de flux du point de vue statique, de problèmes et de tâches du point de vue dynamique.

Le module de visualisation exploite toutes les sortes de relations entre les variables et s'appuie sur des méthodes élaborées de tracé, dont deux sont nouvelles : la première s'inspire d'exemples a priori comme dans le raisonnement à base de cas, la seconde utilise les notions de monotonie et de concavité pour déduire des lignes dans un ensemble de points.

L'application est exposée dans le détail et conduit à une analyse des problèmes de conception, et nous avons développé notamment une nouvelle classification de ces systèmes.

Mots-clés : tâches, courbes, conception, procédés de raffinage

ABSTRACT

A help environment based on a task architecture and a curve visualization module. Application to the conception of refining processes.

In the field of chemical engineering, engineers trace curves to analyze experimental data. Once validated this knowledge is exploited, in combination with other information, in the form of tasks. This thesis presents an architecture capable of linking any kind of task and to draw the curves as a conceptual aid in the design of refining processes.

The proposed architecture is based on an 'object' analysis of the reasoning mechanism including the concepts of relations (inversible or not) and flux from a static point of view and the notions of problems and tasks from a dynamic point of view.

The visualization module employs all sorts of relations between the variables and relies on elaborate interpolation methods, of which two are new. The first is inspired by *a priori* examples as in case-based reasoning, and the second uses the notions of monotony and concavity to deduce lines within a group of points.

The application is exposed in detail and leads to an analysis of the problems of conception. In this sense we have developed a new classification of the conception systems.

TABLE DES MATIÈRES

I) INTRODUCTION

I.1) DESCRIPTION DES PROBLÈMES

I.2) PRÉSENTATION DE LA THÈSE

I.3) ORGANISATION DU DOCUMENT

II) CONTEXTE ET PROBLÈMES

II.1) INTRODUCTION

II.2) LE DOMAINE
- II.2.1) Les Procédés
- II.2.2) Génie Chimique et bailleur de procédé
- II.2.3) Les charges et les conditions opératoires
- II.2.4) Les coupes pétrolières
- II.2.5) La distillation

II.3) L'EXPERTISE
- II.3.1) Les appels d'offre
- II.3.2) L'expert procédé et les essais pilotes
- II.3.3) Les corrélations et les abaques

III) UNE ÉTUDE DE CAS

III.1) INTRODUCTION

III.2) PRÉSENTATION DU CAS
- III.2.1) Les données
- III.2.2) Quelques explications et rappels

III.3) LE POINT DE VUE DE L'EXPERT
- III.3.1) Démarche résumée
- III.3.2) Différentes tâches
  - III.3.2.1) Examen de la charge
  - III.3.2.2) Recherche des conditions opératoires
  - III.3.2.3) Détermination du bilan matière et des qualités produits
  - III.3.2.4) Détermination de la consommation d'hydrogène et de l'exothermicité
  - III.3.2.5) Cas exceptionnels

III.3.3) Conclusion
III.3.3.1) Intérêt du système
III.3.3.2) Sept points difficiles de la démarche informatique

III.4) LE POINT DE VUE DE L'APPLICATION INFORMATIQUE : PRÉSENTATION DE PRINCE
III.4.1) Réponse aux appels d'offre
III.4.2) Fonctions de saisie
III.4.3) Raisonnement suivi dans PRINCE

III.5) LE POINT DE VUE DE L'UTILISATEUR DE PRINCE
III.5.1) Analyse des données de la charge
III.5.2) Résolution du cas
III.5.3) Détail de la résolution

III.6) CONCLUSION

IV) **IA, CONCEPTION ET GÉNIE CHIMIQUE**

IV.1) INTRODUCTION

IV.2) QUELQUES TENTATIVES DE CARACTÉRISATION DES PROBLÈMES DE CONCEPTION
IV.2.1) Simon
IV.2.2) Brown et Chandrasekaran
IV.2.2.1) Présentation
IV.2.2.2) Discussion
IV.2.3) Notre proposition de classification des systèmes
IV.2.3.1) Présentation
IV.2.3.2) Explications
IV.2.3.3) Exemples
IV.2.4) Notre proposition de démarche
IV.2.4.1) Présentation
IV.2.4.2) Discussion
IV.2.4.3) Points Difficiles

IV.3) EXEMPLES DE SYSTÈMES DE CONCEPTION
IV.3.1) Exemples d'applications
IV.3.1.1) Dimensionnement de procédés, PRINCE
IV.3.1.2) Choix de matériel
IV.3.1.3) Formulation
IV.3.1.4) Schémas de procédés
IV.3.1.5) Dimensionnement d'appareil
IV.3.2) À travers des techniques informatiques
IV.3.2.1) Contraintes

IV.3.2.2) Case-Based Reasoning
IV.3.2.3) Tâches
IV.3.3) À travers le domaine du génie chimique
IV.3.3.1) Environnement intégré
IV.3.3.2) Environnement dédié

IV.4) CONCLUSION

V) **ARCHITECTURE À BASE DE TÂCHES**

V.1) LA DÉMARCHE DE RÉSOLUTION
V.1.1) Observation des experts lors de la résolution d'un problème
V.1.1.1) Ils ne procèdent pas de manière séquentielle
V.1.1.2) Les tâches dégagées sont de nature quelconque
V.1.1.3) Les tâches sont parfois inversibles
V.1.1.4) Ils partent des objectifs
V.1.1.5) Les données diffèrent d'un cas à un autre
V.1.1.6) Conclusion
V.1.2) Justification d'une architecture à base de tâches

V.2) TÂCHES DANS PRINCE : DÉFINITIONS ET MISE EN OEUVRE
V.2.1) Définitions
V.2.1.1) Sens des mots tâche et but
V.2.1.2) Représentation statique des connaissances
V.2.1.3) Utilisation dynamique des connaissances
V.2.1.4) Implantation des définitions
V.2.2) Représentation statique des connaissances dans PRINCE
V.2.2.1) Les flux et les relations
V.2.2.2) Différents types de flux
V.2.2.3) Différents types de relations
V.2.2.3.1) Les relations de contrôle
V.2.2.3.2) Méthodes sous-jacentes aux relations
V.2.2.4) Deux conséquences de notre représentation du raisonnement
V.2.2.5) Utilisation de façon isolée
V.2.3) Utilisation dynamique des connaissances dans PRINCE
V.2.3.1) Problèmes
V.2.3.2) Résolution
V.2.3.3) Enchaînement automatique des tâches
V.2.4) Exemple simple
V.2.4.1) Définitions
V.2.4.2) Représentation statique des connaissances
V.2.4.3) Utilisation dynamique des connaissances
V.2.5) Conclusions

V.3) COMPARAISON AVEC D'AUTRES TRAVAUX SUR LES TÂCHES EN IA
V.3.1) Récapitulatif
V.3.2) Tâches et méthodologie
V.3.2.1) Tâches et SADT
V.3.2.2) Tâches génériques
V.3.2.3) KADS
V.3.2.3.1) Présentation
V.3.2.3.2) Discussion sur KADS
V.3.2.4) Synthèse
V.3.3) Quelques applications
V.3.3.1) DSPL, 'Design Structure and Plans Language'
V.3.3.2) SCARP, tâches et résolution de problèmes
V.3.3.3) LISA, opérationnalisation du modèle conceptuel

V.4) CONCLUSIONS

ANNEXE : POINTS DE DISCUSSION

VI) **COURBES**

VI.1) INTRODUCTION

VI.2) ANALYSE DES POINTS EXPÉRIMENTAUX
VI.2.1) Des points expérimentaux difficiles à exploiter
VI.2.1.1) Une grande variété de mesures
VI.2.1.2) Peu de modèles
VI.2.1.3) Des modèles non linéaires
VI.2.2) Étude des cas
VI.2.2.1) Tracer des courbes avec peu de points
VI.2.2.2) Tracer des courbes avec un nombre suffisant de points, mais pas de modèle
VI.2.2.3) Un modèle est disponible, ou posé a priori
VI.2.2.4) Résumé des problèmes
VI.2.3) Quelques solutions
VI.2.3.1) Tenir compte de la monotonie et de la concavité
VI.2.3.1.1) Exposé du problème
VI.2.3.1.2) Solution
VI.2.3.1.3) Algorithme
VI.2.3.1.4) Cas des courbes où seule la concavité est donnée
VI.2.3.1.5) Détail de la génération des lignes
VI.2.3.1.6) Discussion
VI.2.3.2) Compléter des courbes à partir de cas
VI.2.3.2.1) Présentation
VI.2.3.2.2) Démarche

VI.3) EXPLOITATION DES COURBES PAR LES EXPERTS (VALIDATION ET UTILISATION)
VI.3.1) Validation
VI.3.2) Utilisation
VI.3.2.1) Exemple type
VI.3.2.2) Des relations inversibles
VI.3.2.3) Des relations de différentes natures entre les variables
VI.3.3) Mise en oeuvre
VI.3.3.1) Tâches et visualisations
VI.3.3.2) Visualisation des réponses possibles
VI.3.3.3) Analyse des courbes expérimentales
VI.3.3.4) Optimisation

VI.4) CONCLUSION

ANNEXES
VI.A.1) La modélisation des réactions chimiques et la loi d'Arrhénius
VI.A.2) Routine d'optimisation
VI.A.3) Interpolation
VI.A.3.1) L'algorithme de Stineman
VI.A.3.2) Avantages et inconvénients de la méthode de Stineman
VI.A.3.3) Autres méthodes
VI.A.4) Monotonie et concavité

VII) **CONCLUSIONS**

VII.1) PRINCIPAUX RÉSULTATS DE CE TRAVAIL

VII.2) PROLONGEMENTS

RÉFERENCES BIBLIOGRAPHIQUES

LISTE DES FIGURES

schéma type d'un procédé d'hydrotraitement	II.2.1
les différentes coupes d'un pétrole brut	II.2.4
exemple d'une courbe de distillation	II.2.5
une des abaques utilisées	II.3.3
données de l'appel d'offre	III.2.1
la démarche résumée	III.3.1
la charge complétée	III.3.2.1
graphique viscosité-distillation	III.5.1
évaluation de la charge	III.5.1
tâche désazotation	III.5.2
un extrait de procédure	III.5.2
un enchaînement de buts	III.5.3
schéma d'un système	IV.2.3.1
tableau des types de problèmes	IV.2.3.1
notre proposition de démarche	IV.2.4.1
les étapes essentielles	IV.2.4.2
diagramme de flux de données, pour la détermination de la température de réaction	V.2.1.2
représentation d'une inférence à la façon des actigrammes de SADT	V.2.1.2
décomposition d'une inférence à 2 sorties	V.2.1.2
la représentation du raisonnement dans PRINCE, schéma statique	V.2.2.1
hiérarchie des classes de flux	V.2.2.2
classe flux	V.2.2.2
classe flux_valeur	V.2.2.2
hiérarchie des classes de relation	V.2.2.3.2
classe séquence	V.2.2.3.2
classe programme	V.2.2.3.2
classe formule	V.2.2.3.2
classe règle	V.2.2.3.2
classe courbe	V.2.2.3.2
utilisation des tâches de façon isolée, exemple de l'HDN	V.2.2.5
résolution des problèmes dans PRINCE, schéma dynamique	V.2.3
algorithme de résolution	V.2.3.2
utilisation des tâches de façon enchaînée	V.2.3.3
équivalence des notions de relation et de flux employés dans PRINCE avec des concepts d'autres études	V.3.1
exemple de découpage en tâches dans KADS	V.3.2.3.1
exemple de schéma d'inférences dans KADS	V.3.2.3.1
hiérarchie des concepts dans DSPL	V.3.3.1
tâches, méthodes et entités dans SCARP	V.3.3.2
modèle proposé par Isabelle Delouis	V.3.3.3
inférence *'calcul de la température de désazotation'*	V.A.1
graphe d'enchaînement des inférences	V.A.1

graphe d'enchaînement des inférences simplifié V.A.1
graphe d'enchaînement sans circuit V.A.1

points expérimentaux VI.2.2.1.1
réseau de courbes tracées par l'expert VI.2.2.1.1
courbe projetée à VVH = 0,4 h-1 VI.2.2.1.1
courbe de distillation tracée 'à la main', VI.2.2.1.2
à partir des 4 points précédents
courbe tracée dans PRINCE, s'adaptant à un nuage de points VI.2.2.2
courbe d'équation VI.2.2.2
$f(x) = 239813,4\,x^3 - 650787,4\,x^2 + 588693,3\,x - 177504,1$
courbe d'équation VI.2.2.2
$f(x) = 37,85218\,x^{18,21948}$
premier essai d'optimisation VI.2.2.3
second essai d'optimisation VI.2.2.3
troisième essai d'optimisation VI.2.2.3
un exemple arbitraire simple VI.2.3.1.2
comment générer un nuage de points VI.2.3.1.4
où l'on fait passer un nombre exponentiel de courbes croissantes
changement d'axes permettant de rendre la courbe (C) monotone VI.2.3.1.5
algorithme résumé : lissage selon l'allure a priori de la courbe VI.2.3.1.6
un exemple de courbe de distillation VI.2.3.2.1
détermination du point (X, T) à partir de la courbe b, VI.2.3.2.2
dont l'écart E_b est le plus proche de E
courbe obtenue en partant des points (5%, 376,5°C), (70%, 477°C) VI.2.3.2.2
algorithme résumé : interpolation d'une base de cas VI.2.3.2.2
tâche désazotation VI.3.2.1
diagramme HDN = f(TSEJ), VI.3.2.1
pour une charge et une pression données
à différentes températures
diagramme TSOR = f(TSEJ), VI.3.2.1
pour une charge et une pression données
à un certain niveau d'HDN
inversion d'une courbe obtenue par un programme VI.3.2.3
écran de lancement d'une visualisation VI.3.3.1
écran de définition et de commande des courbes affichées VI.3.3.2
contrôle d'une optimisation VI.3.3.4
expression sous forme d'arbre VI.A.2
interpolation de Stineman VI.A.3.1
courbe polynômiale de degré 4 VI.A.3.1
détail de la méthode de Stineman VI.A.3.1
transformation d'une courbe convexe VI.A.4
en courbe convexe et décroissante

CHAPITRE I

INTRODUCTION

I.1) DESCRIPTION DES PROBLÈMES

Cette thèse s'appuie sur la réalisation d'un système à base de connaissances, dénommé PRINCE, développé à l'Institut Français du Pétrole, destiné à aider les experts des procédés de raffinage, à concevoir (et à vendre) des unités industrielles. Il s'agit de mettre en forme les données expérimentales, qui sont la base du savoir des experts et ensuite de les utiliser aussi souplement qu'ils l'attendent, en combinaison avec d'autres connaissances, dans l'élaboration des propositions qu'ils adressent à leurs clients en réponse aux appels d'offre reçus.

1) L'étude de la façon de faire des experts nous a permis de définir une architecture pour notre système, dont la principale caractéristique est la **souplesse d'utilisation**. En effet, même si les experts sont capables de définir une démarche, leur pratique quotidienne est beaucoup moins procédurale. Une même source de connaissance est employée à des fins diverses :
- dans le sens de sa définition (les entrées donnent les sorties),
- dans le sens inverse (on part de la sortie, et de toutes les valeurs d'entrée sauf une, et on cherche ce dernier résultat),
- en combinaison avec d'autres sources,
- pour accomplir tout ou partie de la démarche,
- selon une façon de faire prédéfinie ou non.

2) Les ingénieurs raisonnent de plus **graphiquement.** Dans l'application PRINCE, cet aspect prend énormément d'importance. En fait, il est nécessaire de restituer le plus exactement possible les graphiques, les diagrammes et les corrélations qui fondent les raisonnements menés, quelles que soient leurs formes de définition : programmes, formules, courbes données par des points.

3) Comment les experts s'y prennent-il pour **analyser et valider** les points expérimentaux ? La réponse à cette question est simple, mais pourtant elle est longue à développer : il suffit de comparer les points les uns aux autres. Pour se donner une idée rapide, l'exercice consiste à faire passer, par les points disponibles, des courbes dont on n'a pas nécessairement les équations. Il existe un travail considérable effectué par des générations de mathématiciens autour de ces problèmes d'interpolation et de lissage. L'originalité de notre approche consiste à partir des mêmes informations que les experts, pour résoudre ce problème. Les logiciels de tracé qu'on peut trouver sur le marché n'offrent pas cette possibilité. Pourtant, cette fonction est essentielle au travail des ingénieurs.

4) Les ingénieurs chargés d'établir les propositions disent qu'ils interviennent en conception préliminaire. De quelle façon PRINCE et **conception** sont-ils liés ? Ne doit-on pas plutôt chercher des références du côté du **génie chimique** ? De quelles applications ou systèmes d'intelligence artificielle tirer notre inspiration ?

I.2) PRÉSENTATION DE LA THÈSE

Quelles sont les questions qu'une telle application suscite, et qui dépassent le cadre strict d'une simple réalisation ?

• En premier lieu, nous nous sommes appliqués à caractériser un système d'**Intelligence Artificielle pour la conception.** Nous avons analysé la littérature, suivant plusieurs points de vue successifs. Nous en avons d'abord défini une **classification générale des applications** (dont les problèmes de conception). Nous avons ensuite développé une démarche informatique de résolution des problèmes de conception. Cette réflexion a été ensuite appliquée à notre système, et des exemples tirés d'applications du génie chimique, en priorité.

• En second lieu, nous avons mis en oeuvre une **architecture basée sur des tâches** (ou buts) que les experts se donnent pour résoudre les problèmes qu'on leur pose. Ce type d'architecture a un historique déjà long en Intelligence Artificielle, mais les mises en oeuvre pratiques, aussi souples que la nôtre, capables notamment d'exploiter les capacités d'**inversion des tâches**, suscitent des questions qui demandent réponses, pour que cela marche efficacement. Cette ossature, bien rodée maintenant, est indépendante des procédés de raffinage auxquels elle s'applique, et on le verra, proche d'une architecture de résolution de problèmes. Nous nous sommes attachés à la décrire sous forme générique, réutilisable par d'autres.

• Troisièmement, nos experts ont constamment des problèmes de **tracé de courbes** à travers des nuages de points, qu'il s'agisse d'analyser des points expérimentaux, ou de restituer les informations selon plusieurs points de vue. Nous verrons à ce propos, que l'outillage nécessaire pour leur apporter une assistance est déjà considérable. Au-delà de cet aspect conséquent de réalisation, nous avons apporté **quelques solutions originales** à ces problèmes de tracé de courbes : comment à partir de quelques points, correctement disposés ou non, dessiner des courbes dont on connaît l'allure générale, mais pour lesquelles on ne dispose pas d'équation. Nous avons d'une part mis en oeuvre un raisonnement à base de cas, et d'autre part développé de nouveaux algorithmes de lissage, inspirés de la façon de faire des experts. Ces problèmes de tracé sont d'ailleurs peu spécifiques du raffinage, et notre travail ne demande, en fait, qu'à être continué, pour intégrer encore mieux, la démarche et les a priori des experts.

I.3) ORGANISATION DU DOCUMENT

Ce mémoire est organisé comme suit.

• Le chapitre II **présente** quelques concepts liés au domaine du génie chimique, qui est la discipline sous-jacente à la conception de procédé, concepts nécessaires à la compréhension de ce document. Cette présentation permet d'expliquer dans le détail les exemples que nous avons utilisés dans la suite de ce document. Ce chapitre a été séparé du chapitre suivant, l'étude de cas, pour que des lecteurs avertis, ou peu soucieux d'explorer les exemples dans le détail, puissent l'ignorer et le sauter.

• Le chapitre III introduit une **étude de cas**, qui nous permet de présenter le logiciel, qui est le travail d'application de cette thèse. Nous avons pris l'option de partir des cas pratiques, pour ensuite tenter des généralisations et nous détaillerons successivement :
- la façon dont les experts s'y prennent pour résoudre un cas en l'absence d'outil informatique, et nous soulignerons leurs attentes,
- quel système nous avons mis en place pour y répondre et nous présenterons l'application PRINCE,
- et enfin une session, qui explicitera, à partir du cas précédent, comment utiliser le logiciel.

Ce chapitre fournit des exemples que nous exploiterons dans la suite de ce document.

• Le chapitre IV analyse les problèmes de **conception en intelligence artificielle et/ou en génie chimique**. L'idée qui guide ce chapitre est de trouver des exemples de systèmes dont nous puissions nous inspirer. Nous commencerons par examiner la problématique de la conception en IA. Cette étude se décompose en
- un examen des propositions de l'équipe de Chandrasekaran,
- une caractérisation des systèmes de conception,
- et enfin une présentation d'une démarche de conception.

L'ensemble de ces réflexions nous permet de proposer notre propre point de vue, que nous appliquons à un ensemble d'applications de conception du domaine du génie des procédés, et en particulier l'application PRINCE.

Nous élargirons notre investigation en nous intéressant de façon plus générale aux techniques employés en conception, puis nous présenterons quelques applications marquantes dans le génie des procédés.

• Le chapitre V décrit l'**architecture basée sur les tâches**, que nous avons mise en place. Nous avons pris le parti de nous appuyer d'abord sur notre réalisation, et de l'exposer en détail, avant de la comparer à d'autres travaux. L'exposé de l'application se scinde de ce point de vue, en deux : comment représenter les tâches et comment les utiliser. Mais avant de l'aborder, nous prendrons le soin de définir les mots et les notions que nous employons.

La partie plus bibliographique sur les tâches aura pour finalité d'introduire les définitions que nous avons employées, et de les justifier. Nous évoquerons en particulier les tâches génériques de Chandrasekaran, la démarche KADS, et nous nous situerons par rapport à quelques applications du domaine.

• Le chapitre VI expose notre travail sur les **courbes**. Comme une grande partie de la connaissance est constituée de corrélations, exploitées graphiquement, nous montrerons dans ce chapitre **comment la connaissance est construite, validée puis utilisée**.

Validation et utilisation sont en fait intimement liées : c'est en les affichant que les experts jugent les corrélations. Nous isolons donc deux grandes parties :

- 1) une aide au dépouillement des points expérimentaux, Nous détaillerons d'abord quels sont les problèmes auxquels les ingénieurs chargés de suivre les expériences sont confrontés, puis quelles **nouvelles solutions** nous avons pu apporter. Notamment nous développerons plus longuement des méthodes nouvelles de tracé de courbes. La première s'appuie sur une analyse du comportement des courbes suivant leur monotonie et leur concavité. La seconde exploite une base de cas.

- 2) Une utilisation par les experts des courbes déjà constituées. Les corrélations sont fournies en effet sous des formes bien diverses (formules, courbes définies par des points, programmes), qu'il faut exploiter dans un cadre adéquat.

Enfin, nous présenterons notre réalisation sur les courbes de façon plus approfondie, en insistant sur le fait qu'on peut agir sur les courbes à partir de leur définition ou de l'affichage.

• Le chapitre VII conclue ce mémoire en dégageant les perspectives introduites par notre travail.

CHAPITRE II

CONTEXTE ET PROBLÈMES

II.1) INTRODUCTION

Ce chapitre a pour objet de définir le domaine de l'application informatique qui sous-tend ce travail. Notre objectif est ici de fournir les éléments de compréhension nécessaires à la lecture de ce document.

Le travail d'application de cette thèse, le but de l'application PRINCE, est d'**assister les experts procédés** :
- dans la mise en oeuvre et l'exploitation des connaissances qu'ils ont acquises, pour répondre à des appels d'offre qui leur proviennent des raffineries du monde entier
- dans la construction et la validation des connaissances qu'ils exploitent.

Nous présenterons ce qu'est un procédé chimique qui permet de transformer une charge pétrolière en produits, en la faisant passer sur un catalyseur, sous certaines conditions opératoires. Les produits sont, eux, définis coupes par coupes, et se décrivent par l'intermédiaire d'un certain nombre de qualités.

Nous exposerons ensuite ce qu'est un appel d'offre et de quelles façons les experts acquièrent les connaissances qui leur sont nécessaires pour répondre aux appels d'offre.

II.2) LE DOMAINE

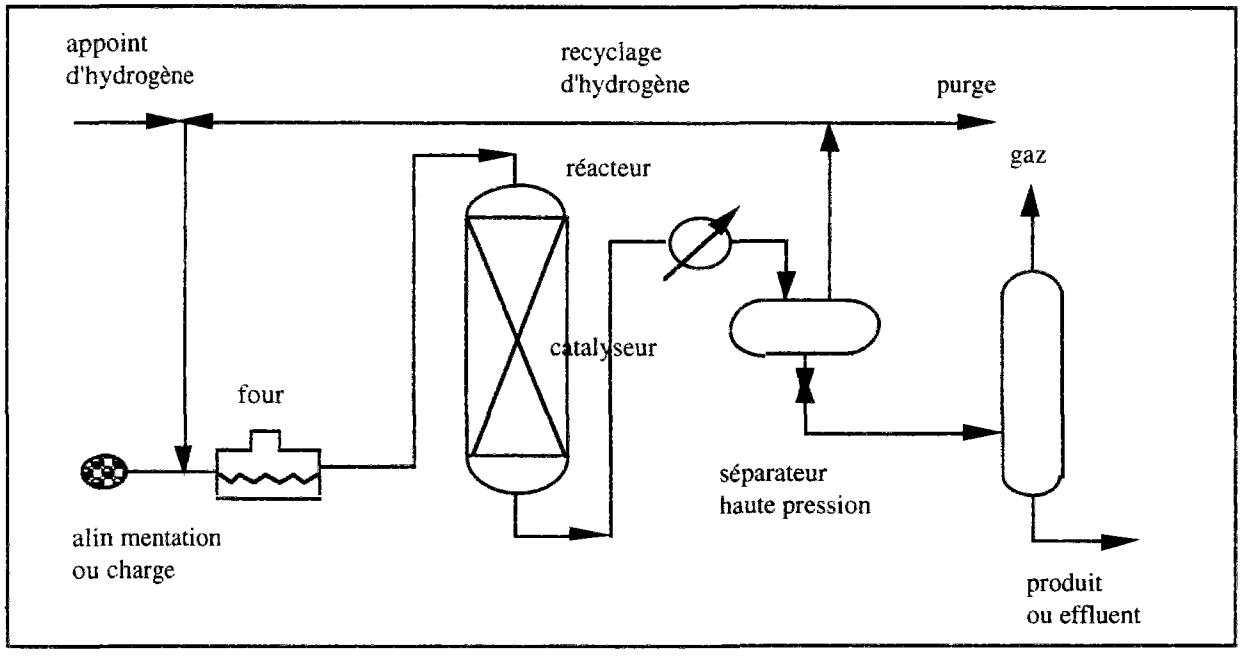
II.2.1) LES PROCÉDÉS

Le but d'une raffinerie est de produire à partir de pétrole brut des produits valorisables ou coupes pétrolières dont les caractéristiques doivent répondre à des normes ou spécifications établies à la fois par les constructeurs automobiles et le législateur.

Le pétrole brut contient une grande variété de composés chimiques qui sont constitués de deux éléments principaux : le carbone et l'hydrogène. On trouve également au sein de certains de ces composés d'autres éléments qui sont souvent considérés comme des impuretés : il s'agit du soufre, de l'azote, de l'oxygène et de certains métaux comme le nickel et le vanadium.

Le fractionnement initial du pétrole brut par distillation permet d'obtenir une douzaine de coupes pétrolières de base. Celles-ci peuvent dans certains cas être utilisées directement pour rentrer dans la constitution des produits finis mais le plus souvent elles sont destinées à alimenter les nombreuses unités de transformation et de conversion que comportent les raffineries modernes : hydrotraitements, reformage catalytique, craquage catalytique, hydrocraquage, viscoréduction...Le schéma 1 présente le schéma d'une unité d'hydrotraitement.

schéma n° 1 : **schéma type d'un procédé d'hydrotraitement**



Le vocable 'hydrotraitement' recouvre toutes les opérations de traitement à l'hydrogène d'une coupe pétrolière, en présence d'un catalyseur solide. On trouve par exemple dans cet ensemble de procédés, les hydrodésulfurations d'essence, de gazole, de distillat sous vide (d'après Trambouze & 84) .

Pour être commercialisée, chaque coupe doit répondre à un certain nombre de **spécifications**, établies internationalement pour certaines, discutées dans chaque pays pour d'autres. Par exemple, les gazoles 'moteur', pour être aptes à la combustion dans un moteur diesel doivent avoir un *indice de cétane* d'au moins 48, les super carburants disposer d'un *indice d'octane* de 98, les kérosènes qui alimentent les moteurs d'avion, montrer un *point d'écoulement inférieur à -50°C* : on imagine mal, en effet, un combustible se solidifier brutalement dans l'aile d'un carburéacteur à 10000 m d'altitude (stage BRP 89).

Les traitements s'appliquent aux coupes et ont pour but :
• l'élimination des composés indésirables sous forme gazeuse (soufre, azote) ou en les transformant en produits non "sensibles", comme dans le cas de l'hydrogénation des aromatiques.
• l'amélioration de caractéristiques figurant dans la spécification (indice d'octane, de cétane, ...)
• la conversion des coupes lourdes (distillat sous vide, résidu, ...) en coupes légères (coupe essence, gas-oil, kérosène, ...) dans le but de répondre quantitativement aux besoins du marché.

II.2.2) GÉNIE CHIMIQUE ET BAILLEUR DE PROCÉDÉ

Le **génie chimique** concerne, dans la dénomination française, le domaine scientifique centré autour des réactions chimiques : comment les déclencher, les entretenir, les réaliser dans les meilleures conditions. Il s'agit de trouver les meilleures conditions d'énergie (température, pression, vitesse), d'environnement (apport d'hydrogène par exemple), de catalyseur qui les favorisent, et de déterminer l'ensemble des moyens technologiques nécessaires à leur réalisation. Le génie chimique est centré autour du **procédé chimique**, l'un désigne le but, l'autre le moyen.

Une fois les réactions déterminées, il faut encore dimensionner les différents appareillages (compresseurs, pompes,...), décider quels sont les matériaux qu'il faut employer en fonction du milieu réactionnel, de la composition des phases, des conditions opératoires (acier au tungstène, par exemple, dont l'épaisseur dépend de la gamme de pression). On parle alors d'**ingénierie** ou de **process** en anglais (terme que l'on retrouve d'ailleurs dans le vocabulaire français).

Une unité chimique coûte de l'ordre de plusieurs centaines de millions de francs, et jusqu'à quelques milliards. Les intervenants, chargés de leur élaboration se partagent le marché en fonction de leur compétence et leur savoir-faire. Celui qui sait élaborer un procédé, comme l'Institut Français du Pétrole (IFP), n'est pas forcément compétent pour faire le plan, au sol de l'unité projetée, ou pour conduire les travaux, domaine réservé des entreprises d'ingénierie. L'IFP intervient comme **bailleur de procédés** (ou bailleur de licence), c'est-à-dire que l'institut opère en vendant la possibilité d'une certaine transformation, et sa maîtrise des règles de mise en oeuvre.

La concurrence entre les différents opérateurs du marché est rude. Leur savoir-faire tient beaucoup aux catalyseurs spécifiques qu'ils élaborent, ainsi qu'aux schémas de fonctionnement qu'ils étudient et mettent au point.

II.2.3) LES CHARGES ET LES CONDITIONS OPÉRATOIRES

L'alimentation d'un procédé s'appelle une **charge**, qui est, on vient de le voir, une fraction non directement commercialisable du pétrole d'origine, fraction qui peut déjà avoir été traitée. Quand une charge n'a pas encore subi d'autres traitements que la distillation (voir schéma n° 2), elle est dite **'straight run'**. A l'issue d'un procédé, chaque produit a quelques caractéristiques bien particulières. Certaines coupes ainsi obtenues demandent encore à être spécialement retraitées.

Pour effectuer les transformations voulues, on fait souvent passer la charge pétrolière à transformer dans des réacteurs chimiques, en présence d'un (ou de plusieurs) catalyseur(s). C'est le catalyseur qui est le coeur du procédé, et c'est lui qui en conditionne bien des aspects. Quand le catalyseur est neuf, le catalyseur est dit '**start of run**'. En fin d'utilisation du catalyseur, quand celui-ci s'est désactivé ou coké, on parle d'un état '**end of run**'.

Les réactions à réaliser imposent les **conditions opératoires**, c'est-à-dire l'ensemble des conditions de pression, de température et de temps de contact à appliquer, nécessaires au bon fonctionnement du procédé. Le temps de contact, ou le **temps de séjour**, est le temps pendant lequel la charge à transformer est en présence du catalyseur. Ce temps dépend du volume de catalyseur placé dans les **lits** du (ou des) réacteur(s), et du **débit** de charge. Son inverse est la **VVH**, pour 'vitesse par volume et par heure', ce qui correspond au débit volumique de charge par heure divisé par le volume de catalyseur.

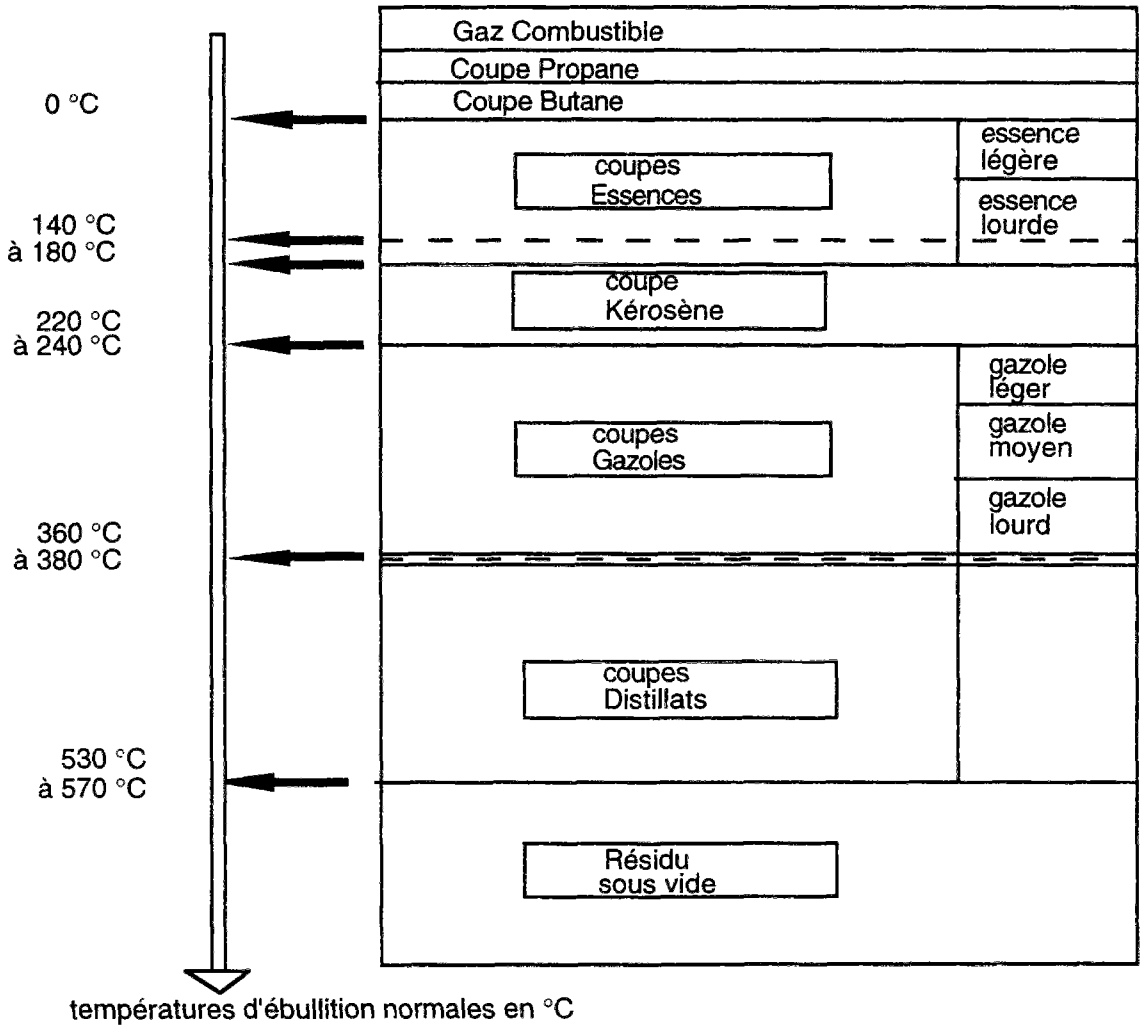
Le temps de séjour conditionne évidemment les réactions à réaliser. A la limite, un temps de séjour infini implique que tout ce qui pouvait réagir a effectivement réagi. Mais si le temps de séjour est infini, le débit est nul. Il faut donc trouver en général le temps de séjour minimal qui permet de transformer tout le débit. De même, dans les réactions d'hydrogénation, la pression d'hydrogène est un facteur favorisant. Malheureusement, plus la pression est élevée, plus le procédé coûte cher, parce que cette condition conduit à augmenter la taille des équipements (la taille du compresseur, la nature de l'acier et l'épaisseur des parois). La température a également un effet positif, dans certaines limites. L'art de la personne qui conçoit un procédé est donc de trouver le meilleur équilibre entre tous ces facteurs, compte tenu du fait que la marge de manoeuvre est étroite, et que la concurrence est vive entre tous les bailleurs.

II.2.4) LES COUPES PÉTROLIÈRES

Les différentes notions exposées dans ce paragraphe sont issues de (stage BRP 89).

Le premier traitement subi par les pétroles bruts dans les unités de fabrication d'une raffinerie est un fractionnement, essentiellement par distillation, qui permet d'obtenir une douzaine de **coupes pétrolières**, ou fractions élémentaires (voir schéma : les différentes coupes d'un pétrole brut), dont les volatilités se rapprochent de celles des produits commerciaux. Ces coupes vont ensuite être retraitées par l'intermédiaire des procédés chimiques de la raffinerie, dans le but d'améliorer leur qualités et/ou de les convertir.

schéma n°2 : **les différentes coupes d'un pétrole brut**



Chacune des coupes pétrolières obtenues par distillation correspond à un intervalle de volatilité qui peut être caractérisé par la gamme des températures d'ébullition ou par le nombre d'atomes de carbone des hydrocarbures qu'elle contient. Ce découpage des différentes coupes pétrolières par nombre d'atomes de carbone et par les températures frontières les plus classiques est représenté sur le schéma 2. Selon la raffinerie, ce fractionnement présente de nombreuses variantes et dépend :

- de la nature du pétrole brut traité,

- de l'équipement de la raffinerie en unités de séparation, de transformation et de conversion,
- et du marché des produits finis alimentés par la raffinerie.

Pour décrire une coupe pétrolière, on trouvera d'abord
- une mesure de la distillation (voir section II.2.5),
- de la densité (ou de la masse volumique),
- de la viscosité à une ou plusieurs températures.

Viennent ensuite, suivant les coupes et les procédés auxquels on les destine, des teneurs en constituants parasites des hydrocarbures En effet, un pétrole, quand on l'extrait, contient un certain nombre d'impuretés, qu'il s'agit d'éliminer totalement ou partiellement avant la commercialisation. On trouve dans cette catégorie, le soufre, l'azote, les métaux (nickel, vanadium, en particulier).

Puis, on décrit les compositions des hydrocarbures, par grande famille. Par exemple, on précise le pourcentage poids de paraffine, c'est-à-dire de molécules d'hydrocarbures qui se présentent sous forme d'alcanes linéaires, d'aromatiques, c'est-à-dire des molécules qui contiennent des cycles saturés, etc.

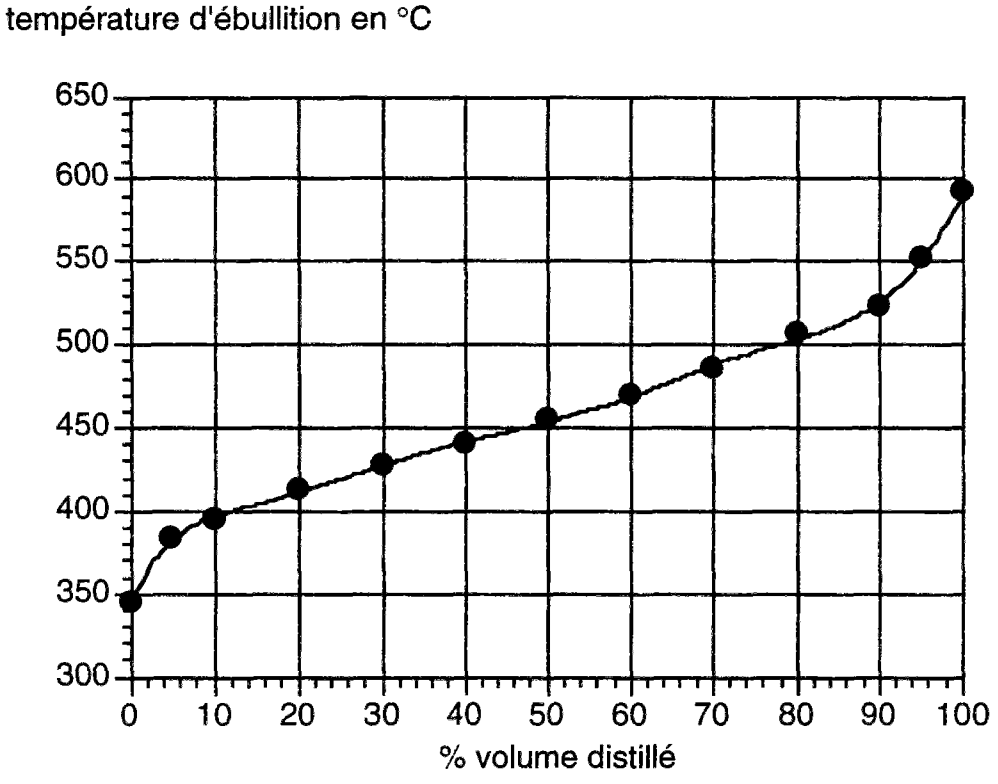
Chacune des coupes pétrolières (essence, gazole, distillat, ...) se caractérise en fait par l'énoncé d'un certain nombre de propriétés, ou de **qualités**. Ces propriétés font l'objet de standards internationaux, recueillis dans les normes de l'API (American Petroleum Institute). Les mesures, ainsi recensées dans les livres de l'API, décrivent minutieusement les appareillages nécessaires et la façon de procéder.

Ces mesures sont parfois relativement simples ou rudimentaires, parce qu'il faut s'assurer que toutes les raffineries, quelque soit leur situation géographique, puissent les mener à bien. C'est dire que pour décrire une charge, ou une coupe pétrolière, les informations disponibles sont peu nombreuses. Le coup d'oeil de l'expert reste important pour juger l'ensemble de la coupe.

II.2.5) LA DISTILLATION

Exploitant les différences de volatilité des constituants d'un mélange, la distillation (voir schéma n° 3) permet la séparation ou, comme l'on dit généralement, le fractionnement de ces constituants en fonction de la température d'ébullition (Wuithier 72). La correspondance entre intervalles de température d'ébullition et rendement offert par un pétrole brut est obtenue par l'analyse TBP (stage BRP 89). Le résultat essentiel de l'analyse est traduit par la courbe TBP du brut ou courbe reliant les températures d'ébullition relevées aux quantités distillées.

Schéma n°3 : **exemple d'une courbe de distillation**



Exemple de courbe de distillation TBP
d'un distillat sous vide

Il existe plusieurs méthodes d'analyse pour obtenir une courbe de distillation. Certaines d'entre elles font l'objet de normes internationales, dites normes ASTM.

II.3) L'EXPERTISE

II.3.1) LES APPELS D'OFFRE

Les sociétés de raffinage exploitent des procédés, que des centres de recherche tels que l'Institut Français du Pétrole conçoivent et mettent au point. Dans ce cadre, l'IFP a un rôle de bailleur de licence (ou de bailleur), c'est-à-dire que l'institut cède les droits d'utilisation des procédés qu'il élabore, en échange de royalties.

Deux cas d'intervention sur le procédé vont être soumis au bailleur : d'abord, la **conception d'une nouvelle unité**, et ensuite l'examen (suivi éventuellement de son **remplacement**) d'un catalyseur déjà installé, parce qu'il faut le renouveler après usure, qu'il ne donne pas satisfaction, ou parce que les conditions d'utilisation ont changé.

Pour répondre à ces demandes, les bailleurs de licence disposent à la base de peu d'informations. Ces dernières sont contenues dans un **appel d'offre** sommaire (voir étude de cas), qui typiquement se décompose comme suit :
• d'abord figurent des données sur la **charge** que l'on souhaite transformer. Les industries du raffinage disposent en effet de sources d'approvisionnement privilégiées, censées peu varier dans le temps, et les procédés de raffinage sont dimensionnés en fonction des charges prévues.
• ensuite viennent des informations sur les **objectifs** du procédé projeté. On souhaite par exemple désulfurer un pétrole, ou obtenir le maximum de kérosène à partir de charges lourdes, par l'intermédiaire d'un procédé d'hydrocraquage.
• Puis on précise ce que l'on connaît de l'**unité**. Dans le cas d'un changement de catalyseur, on décrit le schéma de procédé (la taille du réacteur, le nombre et le volume des lits de catalyseur, les boucles de recyclage, ...). Dans tous les cas, on précise les conditions opératoires imposées, ou des informations qui permettent de les déduire. Les fluides externes, comme l'hydrogène d'appoint dans le cas des procédés d'hydrotraitement, sont également précisés.
• enfin, les **qualités** des produits attendus sont décrites coupe par coupe, plus en détail. Quel est le point de fumée de la coupe kérosène, quels sont ses points initiaux et finaux, etc. ?

II.3.2) LES ESSAIS PILOTES

Les réponses aux appels d'offre sont supervisées par un **expert 'procédé'**, spécialiste des réactions chimiques qui doivent avoir lieu. Celui-ci intervient d'abord pour déterminer les conditions générales de fonctionnement de l'unité. C'est lui en fait qui vend l'unité, parce qu'il est capable de dire dans quelle mesure les objectifs des clients raffineurs pourront être atteints, et sous quelles conditions opératoires (pression, température, temps de séjour, débit).

L'expert base son savoir-faire sur ses connaissances du domaine, sur un long passé professionnel et de nombreux contacts avec les clients, qui lui ont permis

d'affronter des situations variées, mais aussi sur des expériences chimiques, dites **expériences pilotes**, parce que menées sur des installations de taille réduite mais représentative des unités industrielles. Ces expériences consistent à mener des essais complets de transformation de charge.

Ces essais permettent d'obtenir un grand nombre et une grande variété de données expérimentales. Ces dernières sont ensuite travaillées et mises sous forme de **corrélations** graphiques, c'est-à-dire sous forme de courbes mais aussi sous forme numérique, c'est-à-dire sous forme de formules analytiques simples, souvent polynomiales. Ces corrélations sont la base indispensable de la production de propositions industrielles, en réponse à des appels d'offre.

En début, au cours, et en fin d'un essai pilote, des mesures expérimentales sont faites. Ces mesures concernent au début la charge qui doit être traitée. En fin d'essai, on analyse minutieusement les produits obtenus. Au cours de l'essai pilote, on suit soigneusement les évolutions des conditions opératoires.

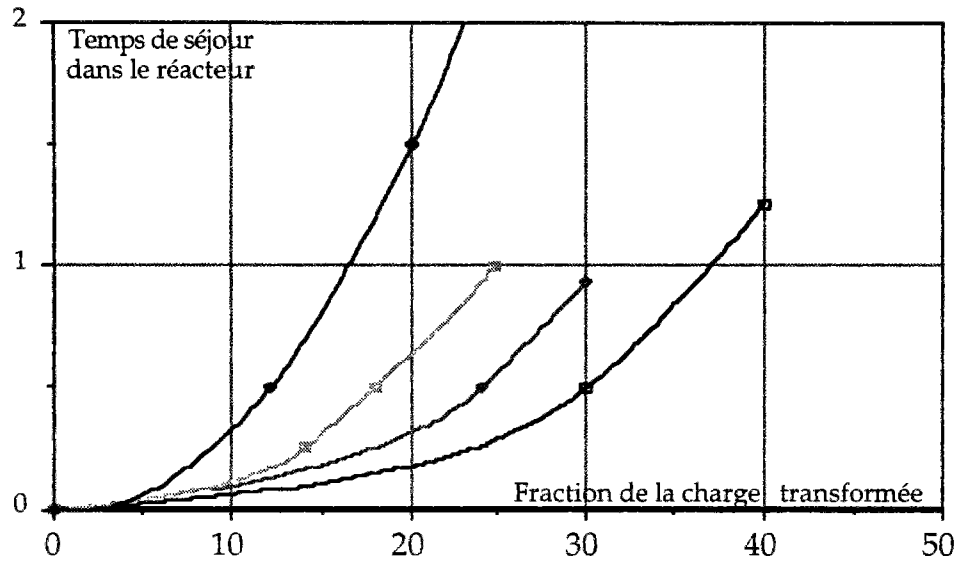
Les mesures expérimentales sont naturellement entachées d'erreur. Du fait de leur grande diversité (on mesure énormément de propriétés, plus d'une centaine), de la grande variété des expériences (ces dernières sont toujours faites sous des conditions différentes), de la difficulté de mise en oeuvre des essais (il peut arriver de rater un essai pilote), du coût financier de telles campagnes de mesure, qui ne permet pas de recommencer s'il y a un doute, l'analyse des données mesurées est un travail ardu.

II.3.3) LES CORRÉLATIONS ET LES ABAQUES

Les abaques sont des représentations analytiques ou graphiques des relations liant les divers paramètres des expériences pilotes. Par exemple, on cherche quelle est la fraction de pétrole transformée dans le réacteur. La quantité mesurée est alors la conversion. Pour identifier la sévérité des traitements appliqués, on utilise les conditions opératoires, c'est-à-dire la pression, la température et le temps de séjour. En balayant le domaine des charges possibles, et l'ensemble des conditions opératoires, on obtient des figures telles que celle qui est présentée ci-dessous (schéma n° 4).

Ces abaques peuvent avec un peu d'effort se mettre sous forme de formule mathématique, sans que cela change beaucoup la nature de la connaissance représentée.

schéma n°4 : **une des abaques utilisées**



Fondamentalement, la charge transformée contient plusieurs milliers de molécules qui réagissent toutes de façon différente. Identifier une seule réaction et la modéliser sous forme cinétique fait déjà l'objet en soi de beaucoup de travail. On imagine donc la distance qui sépare notre capacité de description des réactions et la réalité concrète qu'il s'agit de modéliser. D'autant plus que les phénomènes chimiques se compliquent d'effets hydrodynamiques et thermiques notamment liés à l'écoulement du pétrole dans le réacteur et dans le catalyseur.

Malgré quelques tentatives de modélisation, il est encore nécessaire de construire des abaques expérimentales, seules capables de rendre compte du comportement réel des charges et des catalyseurs.

CHAPITRE III

UNE ÉTUDE DE CAS

III.1) INTRODUCTION

Le but de ce chapitre est de donner des idées concrètes sur l'environnement et l'utilisation du logiciel PRINCE, que nous allons détailler dans la suite de ce document, et qui est le travail d'application de cette thèse.

Dans ce but, nous nous appuierons sur un cas réel, que nous commencerons par présenter.

Puis, nous étudierons comment les experts s'y prennent, en l'absence d'aide informatique, pour traiter un appel d'offre. C'est la partie 'LE POINT DE VUE DE L'EXPERT', qui exploite l'exemple précèdent.

Les conclusions tirées à l'issue de cette partie sont importantes : d'une part, on y justifie l'utilisation d'un système à base de connaissances, et d'autre part, on passe en revue les points délicats, que nous avons été amenés à résoudre dans PRINCE.

Dans la partie suivante, 'LE POINT DE VUE DE L'APPLICATION INFORMATIQUE', nous présenterons alors la partie de l'application PRINCE qui a été construite pour assister les experts dans le traitement des appels d'offre (l'autre partie de l'application sera exposée dans le chapitre VI *courbes*).

Enfin, nous examinerons, à l'aide de l'exemple précédent, comment le logiciel PRINCE fonctionne dans un cas réel, au cours d'une session. Nous développerons à ce niveau un exemple de raisonnement tenu pour répondre à un appel d'offre.

III.2) PRÉSENTATION DU CAS

III.2.1) LES DONNÉES

Cette étude s'inspire d'un cas réel. Cependant, les valeurs des paramètres sont données à titre indicatif, les résultats réels en particulier sont confidentiels. Cette étude concerne un procédé pour lequel il n'existe pas de modèle, on s'appuiera donc largement sur des corrélations. Elle a pour but le remplacement du système catalytique d'un procédé d'hydrocraquage. Voici l'ensemble des données :

schéma n°1 : **données de l'appel d'offre**

Charge	
densité	0.922 à 15 °C
soufre	2.6 % poids
azote	1100 ppm
CCR (carbone conradson)	2.5 % poids
Distillation	D1160
Point 50%	454 °C
Point 70%	493 °C
Point 90.%	543 °C
Point PF	582 °C (Point Final)
Process	
Débit de charge	25000 à 28000 BPSD
Nombre de trains de réacteur	1
Nombre de réacteurs	2
Réacteur N° 1	
lits	2
volume total	117 m3
Réacteur N° 2	
lits	3
volume total	177 m3
Pression d'entrée	100 kg/m2
recyclage gaz	
débit	712 m3/h
pureté en H2	80 % poids
appoint	
débit maximum	178 m3/h
pureté en H2	96 à 100 % poids
Objectif	désazotation
durée de cycle	24 mois

III.2.2) QUELQUES EXPLICATIONS ET RAPPEL

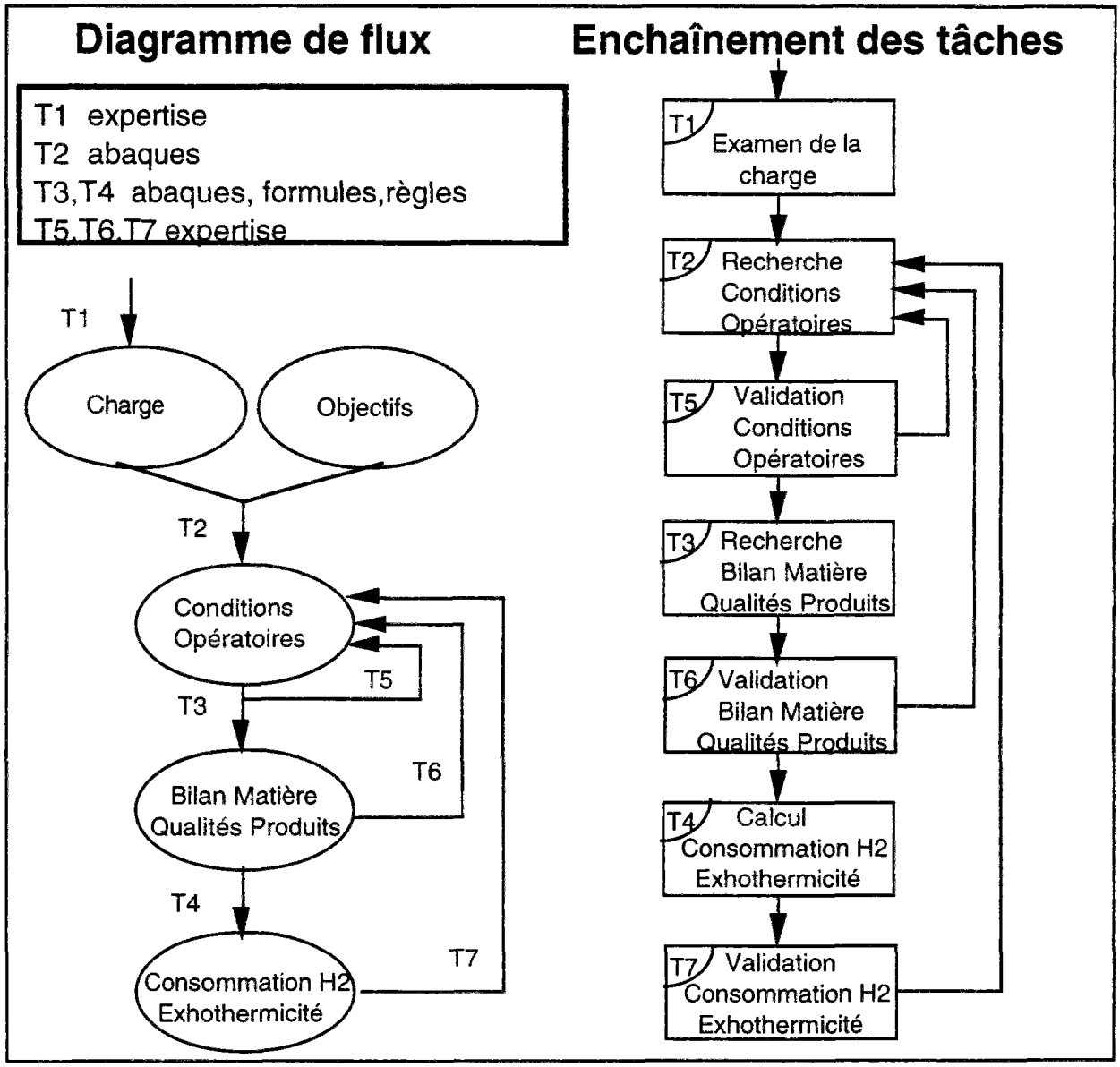
• D1160 est une norme de mesure de la distillation, applicable aux coupes lourdes.
• BPSD signifie 'barrel per day' (baril par jour). C'est une unité anglo-saxonne traditionnelle qui correspond à 159 m^3/h.
• Le carbone conradson est une mesure particulière du pétrole, qui s'applique aux composants lourds de ce dernier.
• Une pression de 100 kg/m^2 correspond à peu de chose près à 100 fois la pression atmosphérique, dans les conditions normales
• La durée de cycle est la durée de vie demandée au catalyseur. Les raffineurs tiennent à renouveler le catalyseur le moins souvent possible, vu son coût, et la lourdeur de l'opération de changement qui nécessite un arrêt complet de l'unité. (par exemple, un mois d'arrêt d'hydrocraqueur coûte plusieurs dizaines de millions de francs).
• L'objectif de désazotation signifie que l'on souhaite enlever presque tout l'azote de la charge parce que l'azote inhibe l'action de certains catalyseurs.

III.3) LE POINT DE VUE DE L'EXPERT

III.3.1) DÉMARCHE RÉSUMÉE

L'étude de cas qui va suivre, va nous permettre de présenter les problèmes que nous avons été amenés à résoudre. Elle présente dans un premier temps le point de vue de l'expert. La démarche suivie est résumée, **pour une lecture rapide**, dans le schéma n° 2, qui comprend un graphe enchaînement des tâches et un graphe des flux. Les différentes tâches, présentées ici, seront détaillées dans la suite de ce chapitre.

schéma n°2 : **la démarche résumée**



III.3.2) DIFFÉRENTES TÂCHES

Après avoir présenté les données, nous allons maintenant décrire comment l'expert s'y prend pour répondre à l'appel d'offre ci-dessus.

III.3.2.1) EXAMEN DE LA CHARGE
(tâche T1)

En toute première analyse, les **données** sur la charge sont **très incomplètes**. En particulier, seuls quelques points sont fournis sur la courbe de distillation, et on ne dispose pas de viscosité, alors que c'est une information indispensable pour mener à bien le travail.

Le premier travail va être d'**estimer les données manquantes** sur la charge. Pour estimer la **viscosité**, on aimerait bien se rapprocher d'une charge connue. Malheureusement, on ne connaît pas l'origine de la charge. La seule façon de faire est alors de supposer que la charge est 'straight run' (c'est-à-dire qu'elle est directement issue d'un fractionnement par distillation du pétrole d'origine), et d'utiliser une des corrélations dont nous disposons. On trouve ainsi une viscosité de 43 cst à 50 °C.

Il faut maintenant **compléter la courbe de distillation** ASTM D1160. L'expert place les points à sa disposition sur papier millimétré; il imagine la forme que doit avoir la courbe de distillation à tracer; *par exemple, la courbe peut démarrer plus ou moins bas suivant la charge, et les points figurant dans les données* ; puis il fait passer une courbe d'allure correcte par les points de base, et il étend la courbe à l'ensemble du domaine de définition; il déduit finalement les points manquants, par simple lecture.

On peut maintenant **mettre aux normes** les données de la charge, c'est-à-dire les ramener aux standards de mesure en vigueur à l'IFP, et compléter les données qu'il reste à déterminer. C'est le travail d'un ingénieur du groupe de proposition (et non de l'expert), car il faut utiliser un certain nombre de programmes informatiques, qui exploitent des corrélations pétrolières confidentielles ou de la littérature.

Peu de temps après, suivant la disponibilité de l'ingénieur, la charge est alors complétée (cf. schéma n°3).

schéma n°3 : **la charge complétée**
Les données en gras ont été calculées

CHARGE

densité	0.922 à 15 °C	
soufre	2.6	% poids
azote	1100 ppm	
CCR	2.5	% poids
Distillation	TBP	
0 %	**297**	°C (Point Initial)
5 %	**337**	°C
10 %	**359**	°C
30 %	**408**	°C
50 %	454	°C
70 %	493	°C
90 %	543	°C
95 %	**555**	°C
100 %	582	°C (Point Final)
Viscosité		
à 50 °C	**43**	cst
à 100 °C	**8.2**	cst
Poids moléculaire		
	394	g
Carbone aromatique		
	18.6 % poids	
Indice de réfraction		
à 70 °C	**1.493**	

On essaie de comparer la charge calculée aux charges connues. **Les caractéristiques de la charge sont examinées** une par une. S'il existe des anomalies, l'expert les corrige directement ou contacte à nouveau le client. Dans cet exemple, les données ont des ordres de grandeur moyens et l'expert ne relève pas non plus d'incohérence entre les mesures : tout semble à peu près normal et moyen.

III.3.2.2) RECHERCHE DES CONDITIONS OPÉRATOIRES
(tâches T2)

Compte tenu de la pression totale, du débit d'hydrogène d'appoint et du débit des gaz de recyclage, l'expert estime une pression partielle d'hydrogène (PPH2), en première approximation. En effet, les gaz sont recyclés, et la PPH2 dépend de ces derniers. Or les gaz ne peuvent se déterminer que si on connaît la PPH2, car ils proviennent pour une grande part de la charge, qui, en réagissant, en libère et en produit. Pour déterminer précisément la PPH2, il faut donc procéder par

itération. A la main, le calcul est extrêmement fastidieux. Tout l'art de l'expert consiste à estimer d'emblée une valeur plausible, qui sera ensuite vérifiée numériquement par un programme de calcul spécialisé.

Il faut maintenant calculer le temps de séjour nécessaire pour atteindre la performance. Rappelons que le temps de séjour est le temps pendant lequel la charge à transformer est en présence du catalyseur. Comme le catalyseur se répartit en lits dans le (ou les) réacteurs, déterminer le temps de séjour conduit à définir le volume de catalyseur de chacun des lits. Là encore, à la main, une bonne appréciation au départ évite de refaire des calculs fastidieux. Manuellement, on fait en général un ou deux essais.

La température de réaction (dite température 'start of run', parce qu'en début d'utilisation du catalyseur) se calcule alors par l'intermédiaire d'abaques, c'est-à-dire d'un réseau de courbes. Si la température estimée dans un premier temps est trop élevée, on cherchera à la diminuer en augmentant le temps de séjour : on augmente le volume de catalyseur et donc éventuellement le nombre de lits de catalyseur, sinon on diminue le débit de charge : cela signifie une moins-value pour le raffineur et ce n'est pas toujours possible à négocier.

Enfin, le catalyseur a tendance à se coker, c'est-à-dire à retenir en son sein des molécules qui ne contiennent presque que des atomes de carbone. Le coke inhibe les catalyseurs et leur activité diminue. Pour maintenir la performance voulue dans le réacteur, on augmente la température, jusqu'à une certain limite T_L, qui dépend du catalyseur, et de la métallurgie du réacteur Plus la durée de vie (ou durée de cycle) désirée pour le catalyseur est longue, plus la température de départ (dite 'start of run') doit être basse. C'est le dernier facteur qui influe sur le choix des conditions opératoires.

Au cas où la température de départ, T, calculée, nécessaire pour atteindre les performances voulues, dépasse la température T_L admissible en début de cycle, on fixe arbitrairement la température T à T_L, et on **réitère** (**tâche T5**) la démarche de calcul.

III.3.2.3) DÉTERMINATION DU BILAN MATIÈRE ET DES QUALITÉS DES PRODUITS
(Tâche T3)

Le bilan matière est la proportion de chaque fraction, ou coupe, de l'effluent total. Il comprend des gaz, et un effluent liquide, qui, lui, se décompose en produits, comme l'essence, le kérosène ou le gazole (entre autres), et en résidu. Ce bilan se calcule par l'utilisation de **corrélations**, et par l'application de formules. Ce bilan s'obtient quand on connaît l'ensemble des conditions opératoires.

Ceci dit, chaque raffineur a sa propre façon de découper la charge. Une coupe kérosène n'aura, d'un industriel à l'autre, pas les mêmes points initiaux et finaux. L'expert fait toute une démarche de correction des résultats pour se recaler par rapport aux points demandés.

La détermination des qualités des produits se fait suivant une démarche identique. Quand toutes les qualités demandées ne sont pas atteintes, ou quand un arbitrage est nécessaire parce que certaines qualités sont antinomiques, l'expert se donne le choix d'aller modifier les conditions opératoires, pour se rapprocher des desiderata de son client, et des objectifs commerciaux. Ce qui introduit un **nouveau cycle** de calcul (**tâche T6**).

III.3.2.4) DÉTERMINATION DE LA CONSOMMATION D'HYDROGÈNE ET DE L'EXOTHERMICITÉ
(Tâche T4)

La démarche est tout à fait identique à celle de la tâche T3. Consommation d'hydrogène et exothermicité s'obtiennent par l'utilisation de formules et d'abaques.

Enfin, quand tout est déterminé, il faut encore vérifier que le recyclage a été correctement estimé. C'est la tâche d'un programme informatique spécifique (**tâche T7**). Si les résultats ne sont pas validés, une modification des conditions opératoires est de nouveau nécessaire.

III.3.2.5) CAS EXCEPTIONNELS

Quand on ne dispose pas de corrélation, l'expert repart des **points expérimentaux** issus des essais pilotes, qu'il retravaille. Plus précisément, on cherche des résultats expérimentaux qui s'approchent le plus possible du cas étudié, et par extrapolation ou interpolation, l'expert déduit les nouvelles valeurs.

Ce processus d'examen des points expérimentaux est en fait complexe et long. Les points de base doivent être étudiés minutieusement. L'expert se donne la possibilité de les écarter quand il les juge hors des comportements normaux ou hors des bornes admissibles, de les corriger quand il a connaissance d'une règle d'écart. Par exemple, toutes les unités pilotes ne réagissent pas de la même façon, et l'extrapolation des résultats à une unité de taille industrielle ne se fait pas identiquement d'une unité à l'autre. De plus, le comportement d'un pilote peut évoluer dans le temps. Les systèmes de mesure également, ce qui fait que des corrections sont parfois nécessaires pour ramener toutes les mesures à une même base de comparaison. Enfin, le coup d'oeil de l'expert reste indispensable, car ce dernier connaît les ordres de grandeur de chaque mesure et est ainsi capable de juger de leurs pertinences.

III.3.3) **CONCLUSION**

III.3.3.1) INTÉRÊT DU SYSTÈME

Le temps d'expert dégagé par le système PRINCE est important : quand il est effectué sans l'aide de PRINCE, ce travail rapidement résumé peut prendre, en effet, une heure dans les cas faciles, ou deux jours quand les qualités à obtenir sont particulièrement difficiles à concilier, alors que l'application PRINCE traite un cas en quelques minutes. D'autant que les facilités offertes permettent d'augmenter la qualité de la réponse. Et deux types d'utilisation sont attendus : une utilisation en réponse commerciale et une aide au suivi des expériences pilotes.

Les tâches à entreprendre pour résoudre un cas sont de nature quelconque, et ceci justifie l'intérêt porté dans ce cas à un système à base de connaissances. En effet, il faut savoir utiliser des appréciations expertes, des formules, des réseaux de courbes, des programmes, ...

Cependant, l'expert sait habilement se situer par rapport aux charges qu'il a déjà analysées et se corrige ainsi. Il tire parti de ses multiples expériences, pour combler d'éventuelles lacunes dans les connaissances déjà formalisées qui sont à sa disposition. L'expert souhaite donc recevoir une aide de l'application PRINCE, qu'il utilise comme une boîte à outils.

Construire un logiciel tel que PRINCE, oblige à formaliser les connaissances. L'idéal serait d'avoir balayé le domaine expérimental par des essais pilotes de telle sorte que n'importe quel cas d'étude nouveau s'y inscrive. Malheureusement, c'est impossible pour de multiples raisons, qui tiennent au nombre de mesures nécessaires et au temps demandé pour les obtenir. Les experts sont indispensables. PRINCE permet ainsi de dresser l'état de l'art et, en intégrant les diverses informations nécessaires à son travail, PRINCE assure la pérennité et la transmission de l'expertise.

III.3.3.2) SEPT POINTS DIFFICILES DE LA DÉMARCHE EN INFORMATIQUE

• 1) La première difficulté de la démarche consiste à **compléter** les données. Un appel d'offre, nous l'avons constaté, est rarement complet. Pour trouver les données manquantes, les sources d'information sont très hétérogènes : il s'agit de mettre en oeuvre des corrélations issues de la littérature pétrolière (corrélation du calcul du poids moléculaire d'une charge, par exemple), des corrélations développées en propre à l'IFP, des procédures numériques (calcul d'un mélange de charges), des comparaisons avec d'autres charges (cas des courbes de distillation - voir le chapitre VI *courbes*). D'autres fois, on consulte une base de données de pétroles d'origines diverses, et on essaie d'en choisir un qui soit ressemblant, et qui serve de base pour déduire les données manquantes.

• 2) Un second point délicat consiste à **apprécier** la charge, c'est-à-dire la situer par rapport à ses congénères, mais aussi à **évaluer** la difficulté du traitement à venir

(voir la section III.5 *le point de vue de l'utilisateur* du présent chapitre). Dans les cas épineux, l'expert peut même demander à ce que des expérimentations complémentaires aient lieu sur la charge de l'appel d'offre, avant d'avancer des chiffres qui vont le lier de façon trop définitive à son client, car les changer, c'est se déjuger.

Un système informatique, face à ce type de problème, doit pouvoir se déclarer compétent, ou demander de contacter l'expert, s'il y a un risque d'erreur. Il doit aussi attirer l'attention des utilisateurs sur des anomalies d'analyse. Enfin, il doit exhiber les charges qui sont les plus ressemblantes.

• 3) **Les données de départ peuvent être variables.** Suivant les situations de base, il va falloir déterminer d'abord, par exemple, le temps de séjour, parce qu'on connaît les autres conditions opératoires; d'autres fois, c'est la conversion (c'est-à-dire la fraction de charge lourde transformée en charge légère) qui va être le facteur déterminant, alors que normalement ce résultat peut être déduit des conditions opératoires. De telle sorte que le raisonnement doit pouvoir être entrepris dans un sens on dans autre, en partant de certaines qualités considérées comme des objectifs (dépendant des cas à traiter), pour obtenir les autres.

Cependant, l'expert est capable, dans les conditions contraignantes de l'interview (Lunn & 91) de décliner une démarche, valable dans des cas standards ou simplifiés. Si la démarche est connue, il faut pouvoir s'y raccrocher.

Le point 3, ainsi que les deux suivants renvoient au chapitre V *architecture à base de tâches.*

• 4) **Les items de la démarche sont de nature variée** : corrélations pétrolières, programmes informatiques, formules, utilisation d'abaques, règles d'expertise peuvent être employés à un moment ou un autre.

• 5) Toutes les personnes intéressées par cette démarche n'ont **pas les mêmes intérêts**. Ce qui intéresse l'expert, c'est de se faire une idée ponctuelle des résultats, puis de les obtenir in extenso. Les ingénieurs chargés de faire les expériences pilotes (à la base des corrélations utilisées) veulent souvent savoir quelle serait la valeur d'une mesure si on se plaçait dans des conditions données, ou encore souhaitent replacer un résultat obtenu dans l'ensemble des points déjà répertoriés.

De telle sorte que l'ensemble des items de cette démarche doivent pouvoir être **enchaînés**, ou utilisés de façon **isolée.** Quand on les enchaîne, l'utilisateur précise les buts qu'il veut atteindre, et déclare quels sont les résultats qu'il veut voir calculer et apparaître dans la feuille de résultats. En utilisation isolée, un résultat doit pouvoir être rendu, quand on a fourni les données de base d'un item.

• 6) L'expert utilise des **courbes**, pour trouver les résultats. Or, les conditions d'utilisation de ces dernières varient et dépendent de la pression, de la température, du temps de séjour, de la charge, des objectifs. Il faut donc savoir

restituer ces corrélations graphiques selon toutes les conditions de tracé que demandent les experts. Ce point est traité dans le chapitre VI, *courbes*, section VI.3.3, *mise en oeuvre*.

• 7) Par ailleurs, s'il ne dispose pas de courbe de base, l'expert repart carrément des **points expérimentaux** dont il dispose. Or, c'est une gymnastique difficile. Certains points sont erronés, il le sait; d'autres sont faux de façon systématique, c'est-à-dire qu'il faut les corriger de quelques pour-cent pour les rendre valides. Mais quelles sont les expériences qui ont ainsi dévié ? De combien faut-il corriger les points ? Quelles sont les mesures susceptibles d'erreur ? Quelle indication prendre quand plusieurs systèmes de mesure différents donnent des indications divergentes ?

Et puis, comment tracer les courbes, quand elles ne passent pas exactement par les points expérimentaux, soit qu'il y ait trop de points, ou qu'il y en ait pas assez ? Comment les dessiner quand on ne connaît que l'allure du tracé, et qu'il n'y a pas de modèle ? Comment décider qu'un point est aberrant, quel critère utiliser ? A ce propos, il faut se reporter au chapitre VI *courbes*, section VI.2.3, *quelques solutions*.

III.4) LE POINT DE VUE DE L'APPLICATION INFORMATIQUE : PRÉSENTATION DE PRINCE

PRINCE est l'acronyme de PRoposition INdustrielle sous Contrôle de l'Expert. Ce nom est, malgré son caractère artificiel, bien représentatif de la démarche mise en oeuvre dans cette application, et de l'architecture de ce système informatique. A tout moment, l'utilisateur garde le contrôle de l'application, et les questions trop difficiles sont renvoyées à l'expert.

PRINCE est structuré en deux grandes parties :
• 1) l'aide à la **production de propositions en réponse aux appels d'offre**, émis par les raffineurs dans le monde, et
• 2) l'aide aux **dépouillements des essais pilotes** effectués dans le centre de raffinage de l'IFP, ce qui se traduit par un ensemble de visualisations graphiques.

Ces deux étapes participent d'une même logique. La seconde est indispensable à la première, elle permet d'analyser et de valider les connaissances qui seront exploitées en production.

PRINCE, dans ce cadre, fait **le lien entre le développement et l'exploitation industrielle des connaissances sur les procédés de raffinage.**

Nous présenterons ci-dessous la partie de l'application informatique liée au traitement des appels d'offre. Cette partie sera développée pour rendre nos discussions ultérieures sur les tâches aussi concrètes que possible. Tout ce qui concerne l'analyse et la validation des connaissances expérimentales sera traité dans le chapitre VI *courbes*.

III.4.1) RÉPONSE AUX APPELS D'OFFRE, VUE D'ENSEMBLE

Le logiciel PRINCE est d'abord conçu pour aider à répondre aux appels d'offre que l'IFP reçoit, dans le domaine des procédés de raffinage. Son but est de choisir le catalyseur adéquat, puis de calculer les conditions opératoires qui permettent d'atteindre les objectifs fixés par l'appel d'offre, et de déterminer les qualités de produits qui en résultent. PRINCE n'est pas concerné dans cette optique par l'ingénierie du procédé.

Comme on l'a vu dans la partie précédente, quand on examine la démarche des experts face à ce type de problèmes, on isole clairement quelques étapes clefs et aussi quelques façons de faire. Le premier soin de l'expert est d'examiner la charge qu'il va devoir traiter, puis d'évaluer la faisabilité de la demande. Ensuite, à l'aide de diverses simulations et de visualisations graphiques, l'expert élabore sa réponse.

• Pour examiner la charge, un ensemble d'outil a été mis au point. Il faut effectuer les changements d'unité nécessaires, se ramener aux standards IFP sur les mesures, trouver les valeurs manquantes, quand cela est possible, et enfin comparer la charge aux charges connues qui servent de référence. On vérifie ensuite que les valeurs des propriétés de la charge sont cohérentes entre elles. Finalement, on qualifie la charge par rapport au procédé : est-ce une charge classique, ou faut-il contacter l'expert et lui laisser le soin de répondre ?

• Un appel d'offre peut contenir plusieurs possibilités de traitement, que ce soit dans le cas d'une unité neuve ou d'un remplacement de catalyseur. PRINCE permet de gérer plusieurs possibilités concurrentes.

• Ayant précisé tous les attendus de l'appel d'offre, deux modes d'utilisation de l'application sont envisageables : obtenir une idée ponctuelle des réponses possibles, ou bien évaluer plusieurs possibilités complètes, pour finalement n'en retenir qu'une. Pour se faire une idée, PRINCE permet le lancement de tâches isolées, c'est-à-dire le plus souvent un calcul simple, qui n'est pas mémorisé. A l'inverse, quand on désire obtenir un ensemble de résultats et étudier plusieurs cas d'étude, l'application met à la disposition de l'utilisateur un ensemble de procédures modifiables, qui permettent l'enchaînement automatique d'étapes de la démarche. Chaque simulation est bien entendu gardée en mémoire, jusqu'au moment où l'on décide de sauvegarder la session.

Plutôt que de passer en revue l'ensemble du logiciel qui a été conçu et mis en place, nous allons présenter dans les deux paragraphes suivants quelques aspects remarquables du logiciel, qui concernent les fonctions de saisie localisées dans l'interface, et le raisonnement suivi dans PRINCE. Ces aspects seront ensuite concrètement illustrés dans la partie 'LE POINT DE VUE DE L'UTILISATEUR'.

III.4.2) **FONCTIONS DE SAISIE**

Ce n'est qu'après avoir étudié finement les données dont il dispose que l'ingénieur cherchera à construire concrètement une proposition de réponse. La saisie et l'étude des données, localisées dans l'interface, fait l'objet d'un ensemble de fonctions distinctes, sans beaucoup de liens les unes avec les autres, dont les objectifs essentiels sont de trouver les données manquantes, de les ramener à des normes connues et de les évaluer.

Trois des fonctions présentes au niveau de l'interface méritent d'être signalées :

- 1) les fonctions qui servent à compléter les données et trouver les informations manquantes.

Certaines sont bien connues dans le milieu du raffinage, et issues de la littérature pétrolière. D'autres utilisent des corrélations développées en propre à l'IFP. Quand on connaît l'origine de la charge, certaines données peuvent être retrouvées par l'intermédiaire d'une base de données de pétroles bruts dont l'IFP dispose.

Un autre moyen de trouver les données manquantes consiste à comparer la charge à des charges connues et d'interpoler les valeurs ainsi découvertes. Pour cela, on utilise les facilités de visualisation des charges, décrites dans l'item suivant.

Enfin, les **courbes de distillation** sont bien souvent incomplètes, quand on les lit sur les appels d'offre. Nous avons développé une **méthode originale** pour trouver les points manquants, qui consiste, pour résumer, à s'inspirer de courbes de distillation des charges que l'on considère comme proches de la charge à étudier. Une plus longue explication sera faite ultérieurement dans le chapitre VI *courbes*, section VI.2.3, *quelques solutions*

- 2) les fonctions de visualisation, qui permettent de situer la charge de l'appel d'offre par rapport à des charges connues, complètement disséquées dans des études précédentes. Ces graphiques permettent aux utilisateurs du logiciel de se faire une idée qualitative de la charge, idée qu'on peut ensuite comparer au raisonnement expert mené par le système, et de retrouver des valeurs manquantes parfois.

Ces graphiques sont basés sur le principe suivant : on porte en abscisse une première qualité de la charge, en ordonnée une seconde et dans cet espace à deux dimensions, on trace l'ensemble des points correspondants aux charges connues, ainsi que le point correspondant à la charge de l'appel d'offre (voir à ce propos le schéma n° 4, de la section III.4.1 intitulée *analyse des données de la charge*, du présent chapitre).

- 3) les fonctions d'évaluation de la charge. Ces dernières sont réparties en deux groupes :

D'une part, il s'agit de **vérifier la cohérence des données** entre elles. Les charges d'appel d'offre peuvent présenter certaines anomalies, dues notamment aux imprécisions de mesure, ou à leur caractère rudimentaire. Pour cela, on s'appuie largement sur les visualisations précédentes, dépouillées en présence des experts, dont les commentaires sont réécrits sous forme de règles, après avoir transformé les données numériques en équivalent qualitatif. Une anomalie détectée peut conduire à stopper tous les raisonnements ultérieurs.

Voici un exemple de règles : *si le point final de distillation est élevé, et que la densité n'est pas forte ou très forte et que la viscosité n'est pas forte ou très forte, alors signaler une anomalie.*

D'autre part, on souhaite **évaluer la charge par rapport au procédé.** Certaines charges peuvent se situer aux bornes du domaine exploré systématiquement pour un procédé donné et donc demander le coup d'oeil de l'expert. Par ailleurs, le simple examen de la charge permet d'estimer la difficulté du traitement à venir. Pour illustrer ces points, il faut se reporter au schéma n° 5, de la section III.4.1 intitulée *analyse des données de la charge*, du présent chapitre.

III.4.3) RAISONNEMENT SUIVI DANS PRINCE

Les raisonnements sur le procédé, quant à eux, s'appuient sur des données mises aux standards en pratique dans l'entreprise. Puisqu'il s'agit de raisonnement, au contraire de l'interface, les étapes de calcul peuvent à la demande s'enchaîner les unes avec les autres. Il s'agit pour l'essentiel de suivre le schéma de procédé, pour produire les résultats attendus.

L'analyse fine de la démarche des experts, quand ils résolvent un problème, a conduit à identifier des étapes élémentaires . Chacune d'elles a pour objet de trouver un nouveau résultat numérique, pertinent au niveau du procédé.

Trois types de raisonnement sont offerts.

• Un **raisonnement ponctuel** : connaissant les données d'entrée d'une étape de raisonnement, on en déduit le (ou les) résultat(s). C'est par exemple l'objet de l'étape nommée : *désazotation, courbes d'HDN : Si l'on connaît l'objectif souhaité d'azote, et les autres conditions opératoires, on peut déduire la température de réaction nécessaire, si on l'ignore.* Derrière ces étapes de raisonnement, peuvent se cacher des méthodes de résolution complexes. C'est le cas de la tâche citée, elle fait appel à des méthodes d'interpolation élaborées.

Des détails sur les raisonnements ponctuels sont donnés dans le chapitre V *architecture à base de tâches,* section V.2.2 *représentation statique des connaissances dans PRINCE* .

• Un **raisonnement enchaîné** : il s'agit de mettre bout à bout plusieurs raisonnements ponctuels. Le lien se fait par les données. *A partir de la connaissance de la charge et des conditions opératoires, on peut déduire le niveau de soufre dans l'effluent, après la réaction de désulfuration. Si on connaît ce dernier, on peut déduire la consommation d'hydrogène due au soufre, à l'aide d'une formule simple. On peut reconduire des raisonnements analogues pour chaque type de réaction, et recalculer la consommation d'hydrogène globale nécessaire, qui est une information déterminante* .

Les raisonnements des ingénieurs sont, d'après notre observation dans ce contexte précis, constitués d'étapes élémentaires, significatives pour eux. C'est cette suite d'étapes que nous cherchons à reproduire, parce que nos utilisateurs savent l'interpréter (David & 90). A cet égard, l'application PRINCE peut être qualifiée d'environnement de résolution de problèmes (Rousseau 88).

Toute la question est bien entendu de reconnaître ces étapes à travers l'analyse. Chaque raisonnement doit être décomposé pour cela à son juste niveau et masquer tous les échelons intermédiaires, en particulier les calculs et les mises en oeuvre informatiques. Ce sont en fait les experts qui peuvent indiquer les informations significatives pour eux

Cette partie est développée dans le chapitre V *architecture à base de tâches,* section V.2.3 *utilisation dynamique des connaissances dans PRINCE.*

• Une **facilité d'interprétation graphique** : qualifier cette possibilité de raisonnement serait ici un peu exagéré, quoique les ingénieurs déduisent énormément d'informations des représentations graphiques.
Cette partie graphique est essentielle dans l'application. A partir d'une allure de courbe, les ingénieurs vont savoir s'il faut augmenter un paramètre (la pression, la température, ...) pour atteindre les performances, et dans quelles limites.

Nous n'avons pas cherché à exploiter qualitativement ces graphiques, parce que si l'information qui y est contenue est importante, elle est difficile à manier. L'expert peut, entre autres, très bien s'ajuster en fonction de ce que souhaite le client ('*ceux-là préfèrent travailler à haute pression* '), agir par sécurité ('*cette charge me parait difficile, je me rapproche autant que possible des cas connus* '), ou tenir compte des contraintes économiques ('*je cherche à baisser autant que possible la pression, pour baisser la taille du compresseur nécessaire et le coût des installations* ').

L'affichage des courbes est détaillée au chapitre VI *courbes*, section VI.3, *exploitation des courbes par les experts.*

III.5) LE POINT DE VUE DE L'UTILISATEUR DE PRINCE

Nous reprenons le cas présenté précédemment et nous allons le traiter avec PRINCE. La discussion qui va suivre examine comment l'ensemble des fonctions développées concourent toutes à satisfaire l'utilisateur. Comme précédemment, seuls deux points plus précis sont détaillés ici : les fonctions qui analysent la charge, et le mécanisme de résolution suivi dans PRINCE. Les fonctions de visualisation sont exposées au chapitre VI *courbes*.

III.5.1) ANALYSE DES DONNÉES DE LA CHARGE

La première tâche est bien entendu de saisir les données et d'enregistrer ce travail. Puis l'analyse des données de la charge commence.

Il manque énormément de données, il faut trouver les données manquantes.

Aucune viscosité n'est fournie. Il faut au moins en disposer d'une. Si on connaissait l'origine de la charge, on se reporterait dans la **base de données de charges**, (accessible depuis l'application), et on chercherait un brut de la même origine, dont on estimerait les viscosités correspondantes par interpolation à partir des points de coupe.

Comme ce n'est pas le cas, la viscosité est estimée par corrélation, à partir de la courbe densité-viscosité. On trouve 43 cst à 50 °C.

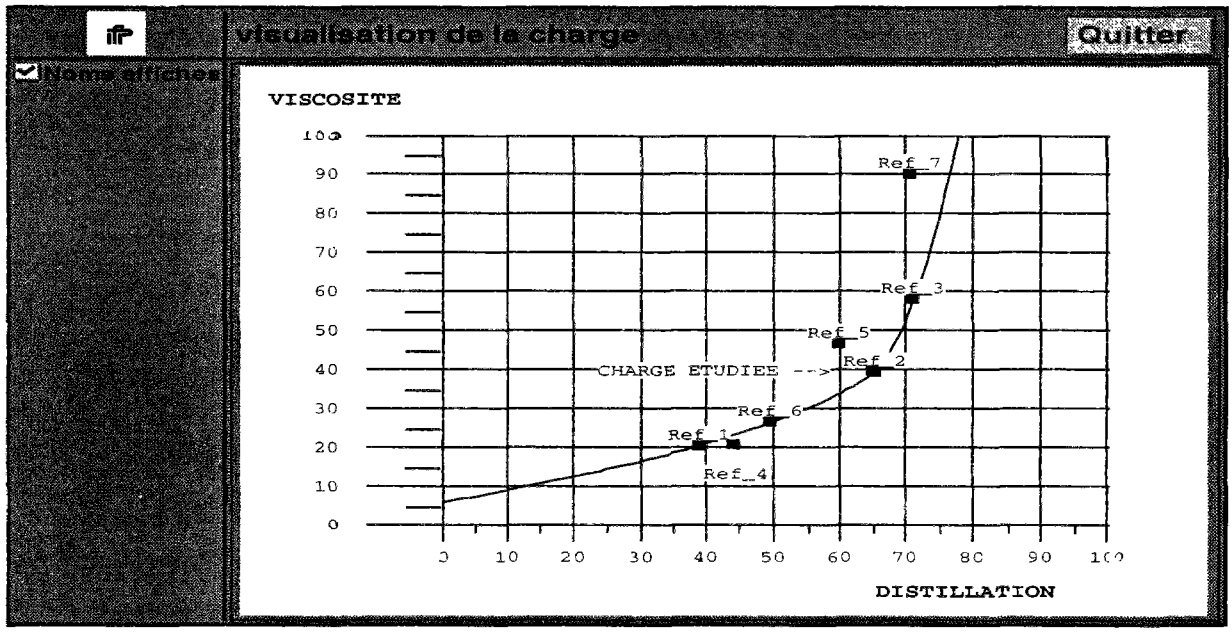
Ensuite, on **complète la courbe de distillation** ASTM D1160 de l'appel d'offre (à partir d'une méthode originale incluse dans PRINCE, voir le chapitre VI *courbes*), puis on la visualise. Manifestement, elle est très plate (c'est-à-dire que la charge est relativement bien coupée), puis on complète les autres données en lançant l'option 'compléter' du menu calcul de l'écran charge. La courbe de distillation ASTM est convertie en TBP, l'indice de réfraction, le poids moléculaire, le carbone aromatique sont **estimés**.

On visualise la courbe de distillation trouvée. Elle a une allure correcte, on l'accepte.

On **visualise les caractéristiques** de la charge une par une (par les graphiques de l'option 'visualiser' du menu 'calcul'). L'idée est de voir si une des caractéristiques choque.

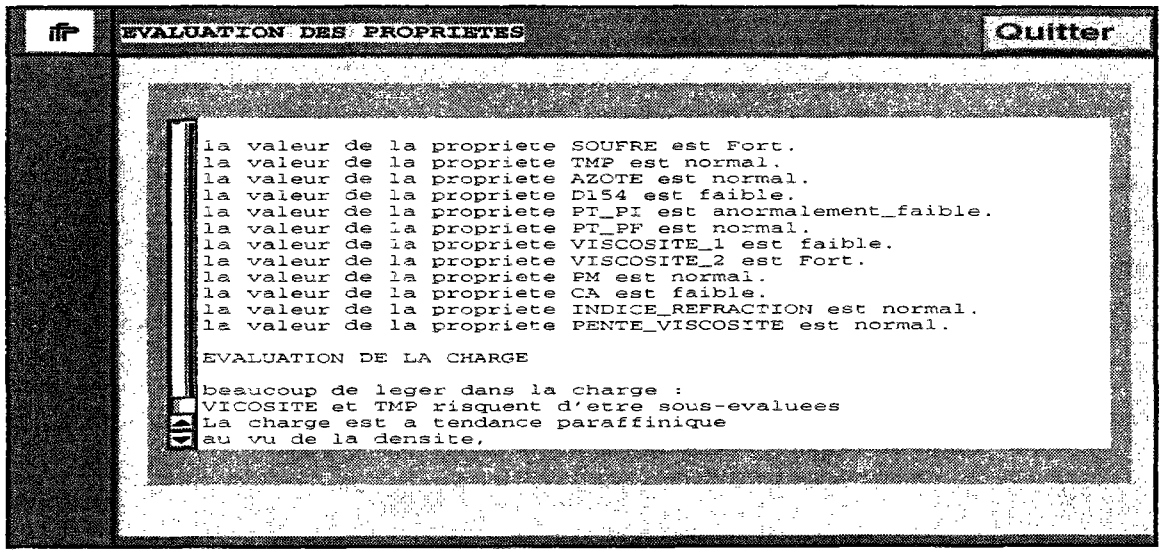
Voici par exemple le graphique *viscosité-distillation (ramenées à une échelle de 0 à 100) La charge se situe exactement sur la courbe moyenne et se confond avec la charge de référence n° 2*

schéma n°4 : **graphique viscosité-distillation**



L'option '**vérifier**' du menu 'calcul' va commenter ces visualisations, de telle sorte que les explications vont faire référence à ces graphiques, qu'on pourra alors afficher à nouveau pour un examen plus attentif. Voila ce que donne l'évaluation sur notre exemple :

schéma n°5 : **évaluation de la charge**

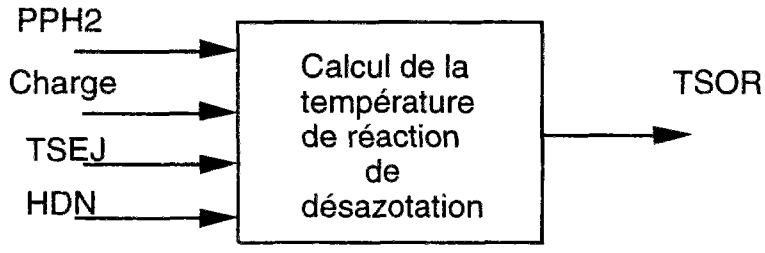


L'option '**caractériser**' regarde comment la charge se comporte vis à vis du procédé envisagé. Elle peut se conclure par 'contacter l'expert', et dans ce cas, les traitements ultérieurs ne sont plus permis.

III.5.2) RÉSOLUTION DU CAS TRAITÉ

A l'aide de la tâche 'désazotation, courbes d'HDN', qui lie les conditions opératoires (PPH2, TSEJ, TSOR) et la charge à l'intensité de la réaction de désazotation (HDN), on va se faire une première idée des températures cherchées.

schéma n° 6 : **tâche désazotation**



Si l'on suppose que les trois lits du premier réacteur sont remplis de catalyseur de désazotation, compte tenu du débit de charge demandé, le temps de séjour vaut TSEJ = volume de catalyseur / débit de charge. La pression partielle d'hydrogène est estimée (PPH2). Le niveau d'azote dans l'effluent étant fixé (HDN), et après avoir choisi un paramètre représentatif de la charge (Charge), on en déduit la température (TSOR) à appliquer à l'aide de la tâche adéquate, que l'on utilise de façon isolée.

D'emblée, cette température parait trop élevée à l'expert. Il faut rajouter un lit de catalyseur. Pour se faire une idée de l'ensemble des couples temps de séjour-température possibles, compte tenu de la pression et de la charge imposées, on visualise la courbe TSOR = f(TSEJ), température 'start of run', fonction du temps de séjour. Implicitement, l'expert se demande s'il parviendra à atteindre les objectifs fixés, et il se donne la possibilité de faire varier un paramètre supplémentaire, le débit de charge (le temps de séjour de la charge sur le catalyseur est fonction du débit de charge).

L'expert va alors explorer certains résultats du traitement à l'aide de plusieurs simulations, en demandant la réalisation d'une suite (prédéfinie) de buts, que nous avons nommée procédure. Chaque cas d'étude qu'il va définir, référencera un traitement différent. Notamment, il essaiera plusieurs valeurs de temps de séjour et comparera les résultats obtenus.

Finalement, l'expert garde les seuls essais qu'il juge intéressants. Une fois l'étude complètement aboutie, elle sera stockée, pour être éventuellement rappelée ultérieurement.

schéma n°6 : **un extrait de procédure**

```
trouver la valeur de la TSOR sur le catalyseur d'hydrotraitement
....
trouver la valeur de la DENSITÉ de l'effluent
....
trouver la valeur du RENDEMENT de la coupe 150-250
....
calculer la CONSOMMATION d'H2
....
```

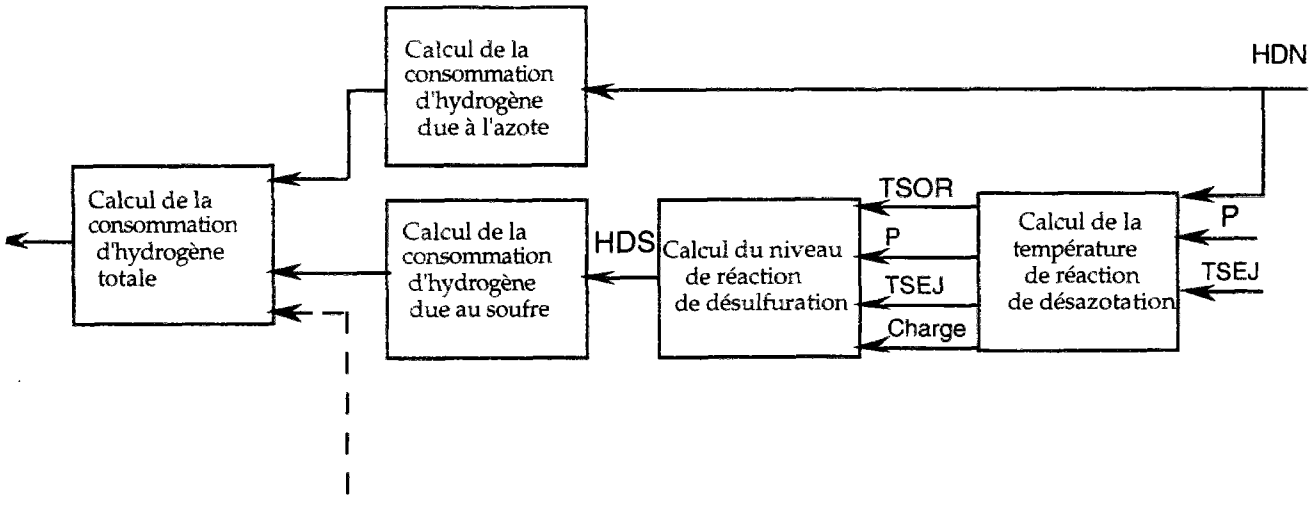
Le système essaie de résoudre les problèmes posés successivement, saute les objectifs qu'il ne peut pas atteindre et propose les résultats pour chaque essai. L'expert juge le meilleur essai (en fonction notamment de ses connaissances commerciales).

III.5.3) DÉTAIL DE LA RÉSOLUTION

Chaque ligne de cette procédure indique un but à atteindre. Si tous les buts successifs sont donnés dans le bon ordre, le traitement que le système tentera sera exactement représenté par cette procédure. Ceci dit, l'expert peut se tromper (par inattention par exemple), mais cela n'empêchera pas l'application de fonctionner. En effet, ce que cherchera à faire le système, c'est de démontrer chaque but, plus que de les suivre mécaniquement.

Prenons un exemple extrême illustré par la figure 7.

schéma n°7 : **un enchaînement de buts**



On demande au système le seul but 'trouver la consommation d'hydrogène', l'application cherchera quelles sont les méthodes à sa disposition pour atteindre

cet objectif, trouvera qu'il lui faut sommer la consommation d'H_2 induite par chaque réaction. Or la seule réaction pour laquelle toutes les entrées sont disponibles, c'est la réaction de désazotation. Repartons de cette réaction de désazotation. Pour obtenir la quantité d'hydrogène consommée ici (CN), il est nécessaire de connaître le niveau d'un paramètre nommé 'hydrodésazotation' (HDN), qui mesure, relativement à la charge, combien d'azote a été enlevé. Ce paramètre est connu, c'est l'objectif du procédé, et on en déduit CN. Le moteur s'intéresse maintenant à une autre réaction, par exemple la désulfuration caractérisée par son intensité, l'HDS. Le moteur identifie que l'un des moyens de connaître l'HDS, est de disposer de la température du réacteur (TSOR). Or, à partir de l'HDN, on peut recalculer la température (TSOR) de réaction nécessaire, puisque l'on connaît ici la charge (C), le temps de séjour (TSEJ) et la pression (P). Avec la température TSOR, on déduit maintenant l'intensité de la réaction de désulfuration (HDS), qui donne à son tour la consommation d'hydrogène due au soufre (CS). Et ainsi de suite.

Notons que les raisonnements conduits s'expriment dans des termes très proches de la façon de penser des spécialistes, c'est-à-dire d'une façon déclarative : les buts ne font pas mention des méthodes employées.

Enfin les buts sont à la disposition des experts, nous leur avons donné la possibilité de reconstruire les procédures (ou suites de buts) qu'ils souhaitent à l'aide d'une interface construite à cet effet. Ceci leur permet d'adapter la marche à suivre en fonction des données disponibles, et des objectifs souhaités.

III.6) CONCLUSION

L'exemple développé ici a pour but principal d'illustrer l'étude plus approfondie que nous allons mener ultérieurement dans ce document.

Le chapitre V *architecture à base de tâches* est consacré à la description de la modélisation et du traitement des connaissances nécessaires pour répondre aux appels d'offre. Ce chapitre répond aux points difficiles 3, 4 et 5 de la section précédente (section III.3.32)

Le chapitre VI *courbes* est plus particulièrement dévolu à la présentation des outils conçus, développés et mis en place pour assister les spécialistes dans le dépouillement des expériences pilotes, c'est-à-dire le construction et la validation des connaissances. Les points 6 et 7 de la section III.3.3.2 sont abordés ici.

CHAPITRE IV

INTELLIGENCE ARTIFICIELLE, CONCEPTION, GÉNIE CHIMIQUE

IV.1) INTRODUCTION

Ayant eu à concevoir et à développer le système à base de connaissances PRINCE, nous nous sommes, tout d'abord, demandés en quoi cette application relevait des problèmes de conception et du génie chimique. Nous avons donc cherché à caractériser PRINCE du point de vue Intelligence Artificielle et Conception, et à situer PRINCE dans les applications d'intelligence artificielle dans le domaine du génie chimique. Nous mènerons ces deux projets en parallèle, plutôt qu'en séquence, c'est-à-dire que les exemples de conception seront plutôt issus du génie chimique et inversement.

Nous commencerons par présenter quelques tentatives de réflexion autour des systèmes informatiques de conception (section 2). H.A. Simon (Simon 69) a justifié l'idée d'une décomposition hiérarchique des problèmes. Brown et Chandrasekaran (Brown & 89) ont introduit l'idée d'une classification des systèmes informatiques en système de conception de routine, innovant et créatif. Nous proposerons alors une classification des systèmes, et notamment des systèmes de conception, en approfondissant une analyse de Clancey. Puis, nous introduirons (comme Tsang (Tsang 91)) une démarche de conception, plutôt qu'une classification.

En section 3, nous illustrerons notre point de vue sur la conception par la présentation de quelques applications en rapport avec le génie chimique, en essayant de montrer comment nos propres idées s'appliquent. A tout seigneur, tout honneur, nous commencerons par caractériser l'application PRINCE.

Nous chercherons ensuite quelles applications sont en rapport avec PRINCE, en élargissant nos investigations aux applications de conception, et à quelques réalisations marquantes dans le domaine du génie chimique. Nous montrerons qu'une grande variété de techniques s'appliquent aux problèmes de conception, et de plus qu'en génie chimique, peu d'applications s'apparentent à la nôtre.

IV.2) QUELQUES TENTATIVES DECARACTÉRISATION DES PROBLÈMES DE CONCEPTION

2.1) SIMON

Le livre célèbre de H.A. Simon, 'The Science of Artificial' (Simon 69) cite les problèmes de conception comme faisant partie des problèmes les plus difficiles que l'informatique et l'intelligence artificielle aient à résoudre. De façon évidente, l'acte de création est bien l'entité qui parait la moins automatisable, puisqu'il s'agit justement d'une pensée qui échappe à des descriptions systématiques, et qui ne se produit qu'une fois quand on résout un problème donné.

Face à des problèmes mal structurés comme ceux de la conception, H.A. Simon propose d'isoler des sous-parties faiblement couplées entre elles, qui permettent de décomposer le problème global en autant d'unités plus faciles à résoudre. Descartes énonçait déjà dans son 'Discours de la Méthode' (Descartes 1637), en parlant des principes qui permettent de bien conduire un raisonnement, la règle de l'analyse, qui consiste à "diviser chacune des difficultés que j'examinerai en autant de parcelles qu'il se pourrait et qu'il serait requis pour les mieux résoudre." Cette idée implique cependant une représentation arborescente de la tâche de conception, qui s'analyse suivant une hiérarchie descendante, puisqu'il s'agit de décomposer. On identifie un sous-système en examinant la façon dont il est relié aux autres, et la règle d'identification consiste à isoler les parties qui partagent le moins d'informations avec le reste.

Une autre idée, développée par H.A. Simon, et largement commentée par Maculet (Maculet 91), est que l'analyse des problèmes de conception conduit à une représentation hétérarchique : une seule représentation hiérarchique ne suffit pas, un objet du champ d'étude fait souvent partie de plusieurs hiérarchies simultanément. Le choix d'un point de vue, inhérent au processus d'analyse conduit inévitablement à des restrictions dans la représentation et le schéma final dégagé tiendra compte de ce que l'on fait des données.

IV.2.2) BROWN ET CHANDRASEKARAN

Brown (Brown & 89) propose une classification, souvent reprise, des problèmes de conception, que nous présentons ici puis que nous discutons.

IV.2.2.1) PRÉSENTATION

La classification de Brown scinde l'activité de conception en trois catégories : la conception de routine, l'innovation et la création.

1) la conception de **routine**, 'routine design', concerne les actions de conception qui ne comprennent pas d'innovation, mais qui tentent l'adaptation à un cas particulier de connaissances et de savoirs connus par ailleurs. Les applications les plus typiques de ce domaine sont les réponses à des appels d'offre, puisqu'il s'agit dans ce type de problèmes, de faire varier et d'adapter des solutions éprouvées et étudiées depuis longtemps. Dans ce type de problème, la structure physique et fonctionnelle de l'objet à concevoir est connue.

2) l'**innovation**, 'innovative design', consiste essentiellement à améliorer par une solution nouvelle, une partie ou un sous-ensemble d'un artefact ou d'un objet. L'ensemble des composants est connu, mais on ne dispose d'aucune méthode directe pour les composer et les assembler, de telle sorte qu'ils satisfassent toutes les contraintes et tous les objectifs d'un problème donné. Il s'agit de transposer une solution connue à un nouvel environnement. La structure fonctionnelle de l'artefact est connue (à quoi sert l'objet ?), sa structure physique n'est que partiellement connue.

3) la **création** 'création' caractérise les domaines où il faut élaborer complètement la réponse, sans qu'on puisse s'appuyer sur des cas connus. La structure des objets du domaine n'est même plus complètement décrite, et encore moins ce que l'on peut faire avec. A la limite, même les objectifs sont à découvrir.

IV.2.2.2) DISCUSSION

Cette décomposition en trois niveaux des activités de conception ressemble à ce qu'on trouve dans l'Encyclopedia Universalis (rubrique Intelligence) : il y est expliqué que trois niveaux d'intelligence peuvent être dégagés : le premier niveau concerne l'abstraction, ce qu'en Intelligence Artificielle, on appelle la classification, c'est-à-dire la définition des concepts. Le second concerne la résolution de problèmes, ce que nous allons montrer comme étant caractéristique des problèmes innovants. Le troisième est le plus difficile, il s'agit de la construction des systèmes formels.

Les exemples abondent de systèmes opérationnels, basés sur une conception de routine. Parmi les premières applications réalisées en conception, on peut citer PRIDE (Mittal § 86) et R1/XCON (MacDermott 82).

Plusieurs exemples existent maintenant de systèmes de conception innovante : Devika Subramanian (Subramanian 93) a produit une application remarquable, briévement décrite au paragraphe IV.3.2.1, qui concerne la conception de systèmes électromécaniques. Siletti (Siletti 90) a construit, pour l'industrie des bio-procédés, un système destiné à déterminer des 'flowsheet' préliminaires, base de revues plus approfondies par les ingénieurs compétents, qui produit des schémas nouveaux en combinant les différentes fonctions et appareillages nécessaires à une transformation donnée.

Il n'existe pas à notre connaissance de systèmes opérationnels de la catégorie 'création'. Et ce constat nous fait penser que trois niveaux de classification ne suffisent pas, il existe certainement d'autres échelons entre innovation et création.

Dans cette classification, conception de routine et innovation supposent un environnement **fermé**, le monde auquel les programmes vont s'appliquer est totalement connu et décrit dans le système informatique. A la base, nous décrivons un objet (à concevoir) comme une combinaison de parties élémentaires, chacune d'entre elles réalisant une certaine fonction. En conception de routine, la combinaison de ces éléments est donnée. Au contraire, c'est l'arrangement nouveau de ces différents composants qui est l'essence de l'innovation : c'est, pour prendre un équivalent, le domaine des systèmes formels, où les axiomes sont connus et en nombre fini.

Par contre, créer, c'est transgresser, c'est-à-dire, apporter des solutions étrangères à l'environnement habituel de travail. Or quel est l'apport essentiel de l'informatique, sinon automatiser un processus soigneusement balisé et analysé. Il y a là une contradiction, qui tient à l'essence même de l'informatique. Quelle est la frontière, entre innovation et création ? Peut-on dire que le peintre ne dispose pas de modèle, alors qu'on sait pertinemment qu'il tire son inspiration de multiples influences ? Inversement, est-il possible de graduer les fonctions d'innovation et de création, pour prétendre caractériser sans ambiguïté ce qui ressort de l'une ou de l'autre ?

Si l'environnement fermé caractérise les développements qui concernent les systèmes de conception de routine ou innovante, les applications de la catégorie 'création' auront lieu dans un **environnement ouvert**. Pour qu'il y ait création, il faut donc que le système interagisse avec le monde (extérieur), qui lui apporte des façons de faire différentes, qu'il s'approprie par analogie, ou par induction, c'est-à-dire par généralisation. Cela suppose de la part du système informatique :
- des capacités d'observation (le monde doit être visible).
- des capacités d'apprentissage, puisqu'il doit augmenter ses connaissances. Qui dit apprentissage, dit structuration des informations perçues et qui dit structuration, dit modélisation. Et quand on parle de modélisation, on peut se demander dans quelle intention, puisque la façon de modéliser dépend de son objectif. Et ainsi de suite... Toutes ces questions ne sont manifestement pas encore résolues.

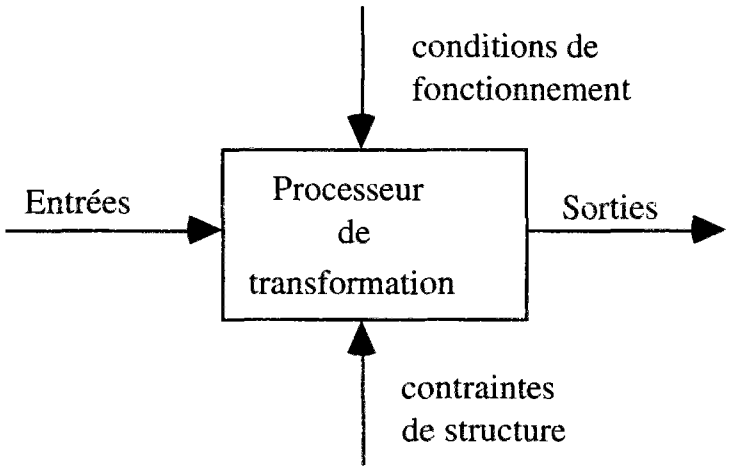
IV.2.3) NOTRE PROPOSITION DE CLASSIFICATION DES SYSTÈMES

IV.2.3.1) PRÉSENTATION

Dans un article célèbre 'Heuristic Classification' (Clancey 85), W. Clancey, repartant des exemples du livre 'Building Expert Systems (Waterman & 1983) propose une classification des systèmes experts de diagnostic et de conception. Pour fonder sa classification, Clancey s'appuie sur des idées issues de l'**analyse des systèmes**, ce qui nous parait un bon point de départ.

Comme le dit par ailleurs Mostow (Mostow 1985), concevoir un artefact, c'est produire une représentation d'une entité physique qui doit remplir une certaine **fonction** en présence de **contraintes** sur les performances, les coûts et les ressources disponibles. De même, un objet dans l'analyse des systèmes se décrit comme un 'processeur' (Lemoigne 1977), néologisme qui signifie qu'il transforme des entrées en sorties. Un objet se définit par sa structure, sa fonction et son évolution (Lemoigne 1977). A la traditionnelle boîte noire (représentant un système), soumise à des informations de contrôle, que nous avons renommées conditions de fonctionnement, nous avons ajouté des contraintes qui s'appliquent à la structure et ignoré les notions d'évolution pendant le temps de l'analyse. Nous arrivons au schéma suivant :

schéma n° 1 : **schéma d'un système**



Par exemple, *un procédé de raffinage est un système, qui a pour entrée une charge pétrolière et pour sorties des produits raffinés, et qui travaille sous certaines conditions opératoires (la température, la pression, ...), et dont le processeur, c'est-à-dire l'agent responsable de la transformation, est, principalement, un réacteur. Une contrainte de structure est le type d'acier employé.*

Un système se décompose en sous-systèmes. Au niveau le plus fin, les systèmes sont dits élémentaires, et un système peut toujours être examiné comme la combinaison de systèmes élémentaires (à la limite, un système élémentaire est

un système composé d'un seul sous-système). Pour définir complètement un système, il faut indiquer, en plus de la liste et de la description des systèmes élémentaires qu'il contient, comment ces derniers sont reliés entre eux, ce qui se fait par la donnée d'un **graphe** (sur lequel existe un ordre partiel) : graphe de flux ou flowsheet pour un procédé, par exemple.

Un système est à un moment d'observation donné, dans un état E, qui est l'ensemble de toutes les valeurs des entrées, des sorties. Cet état peut dépendre du **temps** ou non.

Nous avons précisé notre cadre d'analyse : un système se définit par ses **entrées**, ses **sorties**, ses contraintes de **fonctionnement**, et de **structure**, un **graphe** qui indique quels sous-systèmes il contient et la façon dont ils sont reliés, et par un **état** dépendant du **temps**. Nous pouvons examiner maintenant quels sont les problèmes qui peuvent se poser par rapport à un système donné. Les réponses sont résumées dans le tableau ci-dessous. Les explications sur le détail des types de problème sont données aux paragraphes suivants.

L'idée systématique de ce tableau est de faire varier ce qu'on connaît ou non sur un système donné. Le signe '?' indique que l'information désignée par la case est inconnue, une case laissée en blanc suppose inversement l'information connue. Le 'oui' dans la case 'Temps' signifie que le temps intervient.

schéma n° 2 : **tableau des types de problèmes**

Type de problème	Sorties	Conditions de Fonction[t]	Graphe des sous-systèmes	Contraintes de Structure	État, dépendance /Temps
A	?				non
B		?			non
C	?				oui
D			?		non
E			?		oui
F				?	non
G			?	?	non

Remarques :
- Notre classification porte sur les systèmes et non sur les techniques de résolution employées pour les traiter.
- Les types de problème sont repérés par une lettre de l'alphabet, plutôt que par une dénommination, qui prête toujours à discussion.
- Les termes employés pour désigner les types de problèmes sont génériques
- Les entrées sont toujours supposées connues.
- Toutes les combinaisons des colonnes ne sont pas possibles
- Les problèmes réels sont la plupart du temps des combinaisons de ces problèmes élémentaires.

IV.2.3.2) EXPLICATIONS

• A) Il s'agit de déterminer les sorties, connaissant les entrées et la transformation. Cette activité pourrait être qualifiée d'interpolation ou d'extrapolation, suivant que les valeurs d'entrée sont dans le domaine des valeurs déjà explorées ou non. Les mots de **prédiction** ou encore de **simulation statique** sont parfois employés.

• B) Quelles sont les conditions à appliquer à la transformation T en fonctionnement stable (c'est-à-dire qui ne dépend pas du temps), pour obtenir les sorties désirées ? Clairement, il faut trouver les paramètres qui donnent le meilleur fonctionnnement : il s'agit d'**optimisation**. Nous avons situé les lignes, prédiction et optimisation (A et B), l'une au-dessous de l'autre, car ces deux notions sont en fait très proches. En particulier, l'une des façons de faire, pour optimiser, est de lancer autant de prédictions que nécessaire, en faisant varier les conditions de fonctionnement, jusqu'à trouver l'extremum cherché, s'il existe.

• C) Ici, le temps intervient. Ce type de problème est parfois dénoté simulation ou **simulation dynamique**, et s'attache à modéliser des **états transitoires**. On peut se donner de plus un objectif de contrôle et on parle de contrôle dynamique, et de système réactif.

•D) Il s'agit de trouver la séquence des opérateurs, des transformations qui permet d'arriver au but recherché. Trouver un organigramme des traitements informatiques en est l'exemple type. Dans ce cas, on évoque les problèmes de **plannification** et d'ordonnancement. Ces problème deviennent souvent complexes, quand ils se doublent d'un enjeu d'optimisation : par exemple, dans le cas d'un planning d'atelier, il s'agit d'occuper au mieux les machines.

• E) La séquence des événements qui justifie l'état final observé peut être un élément déterminant. On parle de **diagnostic**. Par certains aspects, le diagnostic ressemble au problème de la planification, puisqu'il faut trouver la séquence des opérateurs qui justifie l'état des sorties observées.

• F) Nous entrons maintenant dans le domaine de la conception. Il faut déterminer quelle est la structure de l'objet à concevoir qui permette de garantir les sorties désirées. Pour l'essentiel, il faut faire un choix parmi différentes possibilités et donner la taille des différents composants. Il s'agit de **configuration**.

• G) C'est la seconde rubrique du domaine de la conception. Ne connaissant pas la séquence des transformations à effectuer, la structure finale de l'artefact est forcément indéterminée. Il est tentant de parler d'**innovation** à ce propos.

IV.2.3.3) EXEMPLES

Comment notre analyse s'applique-t-elle à des problèmes issus du monde réel ? Comme W. Clancey, nous distinguerons les systèmes hors fonctionnement des systèmes en fonctionnement.

Systèmes hors fonctionnement

Nous avons choisi d'illustrer notre propos, pour les problèmes de conception, à partir d'exemples du génie des procédés, et nous nous sommes appuyés sur le cours de simulation en génie des procédés donné à l'école des Mines de Saint-Etienne par P. H. Vacher (Vacher 92).

• 1) Détermination du schéma de procédé
Définition : C'est l'organisation en réseau des opérations unitaires nécessaires à l'obtention des produits désirés aux spécifications requises.
Commentaire : clairement, il s'agit d'**innovation** (G).

• 2) Opérabilité du procédé
Définition : Le procédé est-il stable, peut-on le contrôler ?
Commentaire : dans ce but, on recourt, théoriquement, à la **simulation** dynamique du procédé (C), sauf si l'expérience acquise sur d'autres procédés permet de conclure intuitivement.

• 3) Sécurité opératoire
Définition : C'est l'étude du 'comportement de l'unité hors des limites de fonctionnement normal'.
Commentaire : On recourt, dans ce cas, à la **simulation** dynamique (C).

• 4) Design
Définition : Cette activité comprend deux phases : la première consiste à fixer définitivement le schéma de procédé et les grandeurs opératoires, et la seconde permet, en particulier, le dimensionnement des appareils et leur optimisation.
Commentaire : Dans la première étape (que les gens du génie des procédés appellent conception), il s'agit plutôt de faire un choix parmi des variantes possibles, ce qui en soi est une activité de **configuration** (F), puis de donner des valeurs (les meilleures possibles) aux variables opératoires, ce qu'on a qualifié d'**optimisation** (B), et enfin, quand on les a choisies, de détailler les qualités obtenues, ce qui s'apparente à de la **prédiction** (A). En seconde étape (appelée configuration), on retrouve ce double aspect **configuration** et **optimisation**.

Systèmes en fonctionnement

On range sous le nom générique de diagnostic, un ensemble de problèmes qui touchent, dans le domaine du génie des procédés, à des sujets variés. Le point commun est en fait que les données de départ proviennent d'une unité en fonctionnement. Dormoy (Dormoy 90) cite par exemple le traitement d'alarmes dans les centrales nucléaires, qui consiste à donner la liste des fautes principales

quand une série d'alarmes se déclenchent. De même, le contrôle de procédés a pour mission de suivre une consigne de fonctionnement, c'est-à-dire de faire évoluer une unité industrielle en suivant une évolution connue et cette activité se range aussi sous les termes de supervision (Cauvin & 93), surveillance, ou conduite. Le diagnostic à base de modèles (Raiman 92), quant à lui, se donne pour mission de détecter les composants défectueux d'une machine.

1) diagnostic à base de modèles.
Ici, le temps n'intervient pas. Nous avons classé ce type de problème sous la rubrique **planification** (D). Il s'agit de trouver la séquence de transformations ou d'opérateurs, et finalement de sous-systèmes, qui justifie l'état observé.

2) traitement d'alarmes
Cette activité est en fait très similaire à la précédente, si l'on veut bien considérer que chaque composant défectueux d'une installation envoie une alarme signifiant : 'je suis hors service, ou hors de mes conditions normales de fonctionnement'. Encore une fois, il faut déterminer quels sont les sous-systèmes en faute, et déterminer une séquence de transformations qui explique les alarmes.

3) Contrôle
C'est le domaine de la **simulation dynamique** (C). Pour résumer, il s'agit de détecter des écarts par rapport à une norme, puis de déclencher des actions correctrices. Si un modèle mathématique est disponible, l'outil utilisé est de préférence la simulation dynamique. En l'absence de modèles, ou devant des phénomènes discrets un système expert couplé à un module de prédiction peut convenir. Dans un système embarqué, on parlera de système réactif.

4) Diagnostic
Dans un raisonnement de diagnostic, on conduit un raisonnement du type 1, diagnostic à base de modèles, et on emploie en plus des primitives temporelles, telles que *dès que, tant que, avant, après,* etc. . Les considérations sur le temps compliquent les déductions et influent sur les conclusions dégagées. Notons que le temps est vu ici comme base d'un historique, alors qu'en simulation dynamique, il soutient des analyses de tendance.

IV.2.4) NOTRE PROPOSITION DE DÉMARCHE

IV.2.4.2) PRÉSENTATION

Nous avons examiné précédemment la finalité des applications ou des systèmes de conception, et nous en avons déduit une classification. Une autre façon d'analyser les systèmes de conception est d'examiner comment ils remplissent leur mission. Dans ce but, nous allons proposer une démarche basée sur notre expérience et nos contacts avec les ingénieurs et les experts.

Chaque item de la démarche ne présente pas en soi une originalité majeure. La synthèse présentée ici nous semble cependant intéressante.

Nous partons de l'idée que le problème global de conception a été décomposé en sous-problèmes, une fois pour toutes. Cette décomposition a été principalement effectuée en examinant comment les experts s'y prennent pour traiter les cas qui leur sont soumis.

La première étape, comme l'a bien noté Tsang (Tsang 91), quand on a un objet à concevoir est d'**analyser les spécifications** : sont-elles correctes et complètes ? c'est par exemple une importante fonctionnalité de PRINCE, qui va analyser la charge dans tous ses détails.

La démarche va se poursuivre en traitant les sous-problèmes les uns après les autres et pour chacun d'eux en **générant des hypothèses**, ou des alternatives. Dans des problèmes de configuration, tel qu'il se pose dans la conception de satellites (Trousse 89), *'l'ingénieur est souvent amené à choisir parmi différentes alternatives'* (ch 2, p46), qu'il doit générer et exploiter. Dans un système expert qui fait des choix de technologies, ces différentes possibilités sont recueillies une fois pour toutes, et le système se contente d'exploiter une classification déjà effectuée (Vescovi & 90).

Quand on a généré des alternatives, on cherche à **éliminer les possibilités manifestement inadéquates** (Gondran 93). Quels sont ces critères d'élimination ? Ce sont les contraintes connues à ce moment de la démarche, qu'elles proviennent des spécifications, ou qu'elles soient découvertes au cours de la démarche.

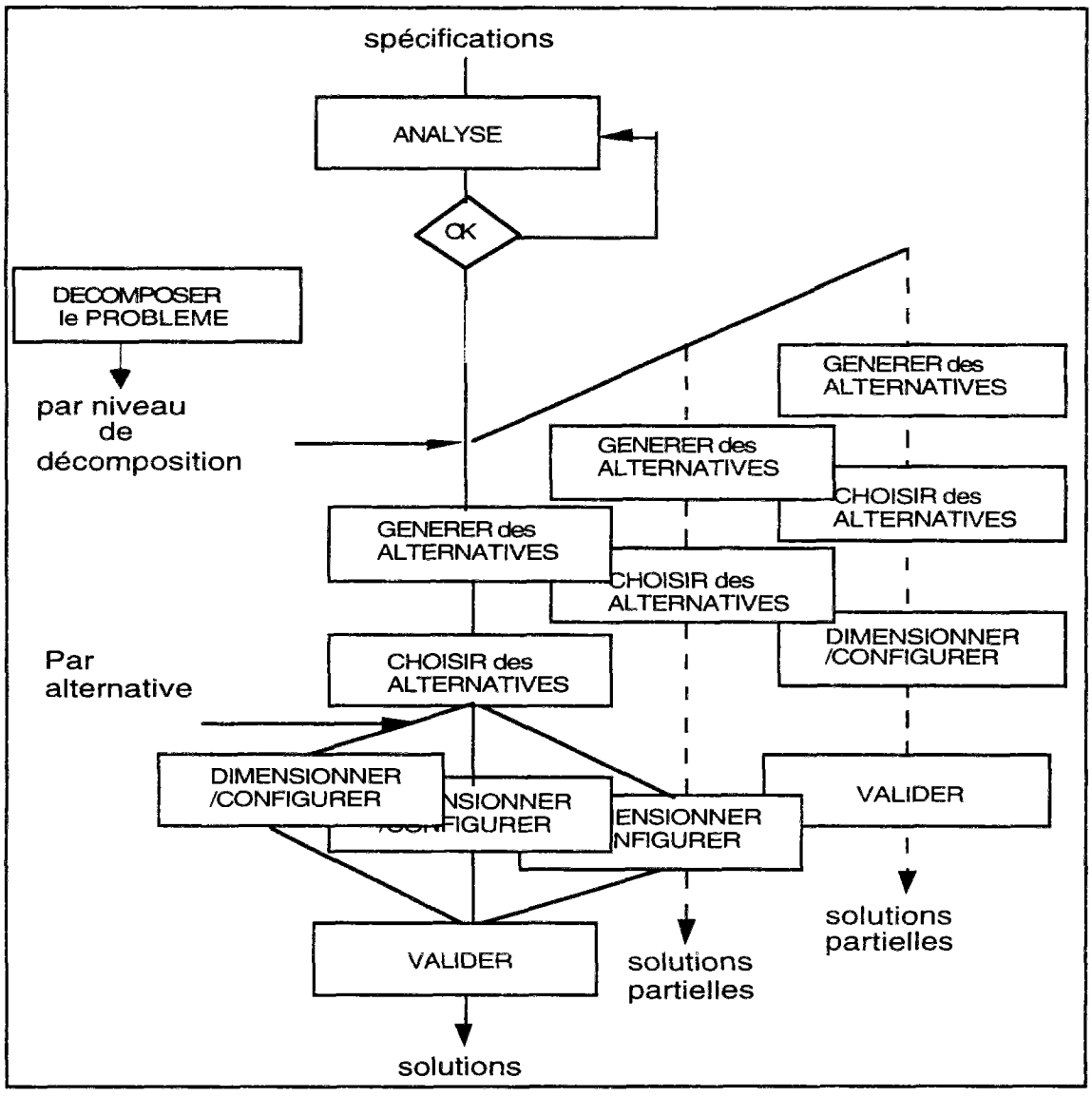
Seulement, rien ne permet de dire si les informations récoltées, au moment où l'énumération des différentes possibilités a lieu, vont être suffisantes pour résoudre complètement ce choix et ne retenir que les hypothèses plausibles. Peut-être faudra-t-il aller jusqu'à l'étape de dimensionnement pour écarter les choix inadéquats, ou peut-être sera-t-il nécessaire de configurer pour prendre les bonnes décisions ? D'autant plus que l'exploration de chaque possibilité peut être coûteuse en ressources d'ordinateur.

C'est pourquoi, après avoir énuméré, il faut **choisir** quelles sont les alternatives qui vont être développées en priorité, parmi les possibilités restantes. Explore-t-on la première branche envisageable, dispose-t-on d'heuristiques qui permettent de privilégier une alternative plutôt qu'une autre, parcourt-on toutes les hypothèses produites, ou demande-t-on à l'utilisateur ce que l'on doit faire (voir par exemple le système LISA, développée par I. Delouis (Delouis 93))?

Pour affiner une possibilité, on peut poursuivre en **configurant** si les contraintes sont d'ordre géométrique, ou en **dimensionnant**, c'est-à-dire donner une valeur, une dimension physique, largeur, longueur, hauteur, par exemple aux propriétés des objets. Dimensionner ou configurer un objet c'est à la fois respecter les contraintes connues et contraindre les sous-systèmes qu'il contient, de telle sorte que le processus de conception recommence, selon un **cycle itératif**, à mesure que l'analyse de l'artefact à concevoir s'affinera.

Enfin, la démarche se terminera par une **étape de validation**, qui ressemble à celle de choix des alternatives.

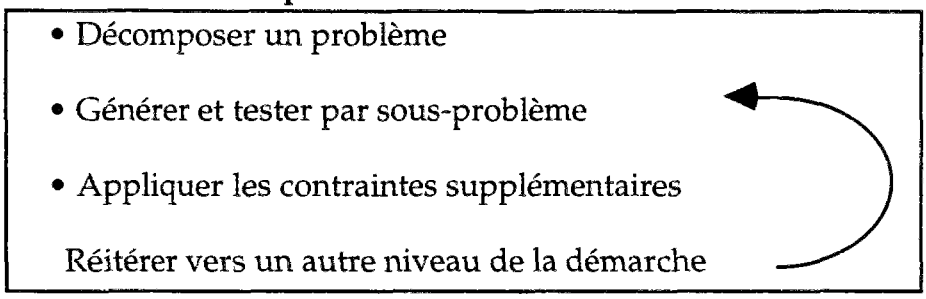
schéma n°3 : **notre proposition de démarche**



IV.2.4.2) DISCUSSION

Nous allons reprendre ici la démarche que nous avons présentée ci-dessus. Le schéma suivant indique les étapes essentielles de façon résumée :

schéma n°4 : **les étapes essentielles**



1) **décomposition en sous-problèmes**

Pour nous, chaque sous-système à concevoir fait lui-même à part entière l'objet d'une démarche de conception. Cette idée de conception appliquée à chaque sous-système rend bien compte du caractère itératif de la démarche que nous avons déjà souligné.

Remarquons également qu'il n'y a pas un mode de raisonnement particulier à recommander dans le domaine de la conception. Comme l'architecture informatique bâtie autour d'un problème, un mode de raisonnement est un moyen et non un but. L'idéal est bien entendu d'établir une architecture qui permette tous les modes de raisonnement.

Cependant, une démarche de décomposition du problème initial doit déjà avoir eu lieu, une fois pour toutes. Rien ne dit comment faire cette décomposition. On rapporte souvent qu'un problème bien posé est déjà à moitié résolu, ce qui signifie, à nos yeux, que poser un problème, c'est prévoir comment le décomposer, même implicitement.

En fait, pour chaque sous-problème, il peut exister plusieurs méthodes, et ce n'est pas toujours la même méthode qui est adaptée. Il y a la méthode ad hoc, à quelque niveau de décomposition que ce soit.

2) **Génération d'alternatives**

En conception, on envisage des alternatives, qu'on va se charger de vérifier ou d'invalider. C'est ce qu'on appelle généralement le paradigme 'generate and test' (Simmons & 87). Pour générer des alternatives, il faut savoir les énumérer, ce qui se fait de façon diverse :

- on peut explorer une classification, une taxinomie déjà établie. On connaît par exemple, l'ensemble des objets qui vont permettre la transformation souhaitée, comme c'est encore le cas pour faire un choix parmi plusieurs appareils.
- on peut explorer une base de cas, comme dans le 'case-based reasoning', ce qui est une façon de faire différente de la précédente. Le problème est alors de trouver par l'application de critères de proximité, les cas jugés les plus proches.

- on peut encore analyser un domaine défini par des contraintes (l'énumération peut être implicite).
- on peut reconstruire les possibilités automatiquement et systématiquement par l'application d'algorithmes spécialement conçus dans ce but, comme dans DENDRAL (Lindsay & 93).

3) **Choix des alternatives**

Pour éliminer les voies manifestement inadéquates, on applique les contraintes connues. L'astuce est de retarder les choix le plus longtemps possible, parce qu'on balaie mieux ainsi l'espace de recherche, et que les critères d'élimination sont d'autant plus efficaces qu'on les applique à un plus grand nombre de possibilités.

Il reste à ordonner les alternatives restantes et faire son choix. L'ordre dans lequel on les essaie n'a pas en soi de conséquence théorique mais il influe de façon déterminante sur le temps de calcul. On suit ainsi l'idée de 'rationnalité limitée' de H.A. Simon (Simon 69), pour se focaliser sur la première solution : certes, ce n'est pas la meilleure, mais comment définir un critère incontestable pour trouver la meilleure solution. De toute façon, c'est une bonne solution, parce qu'elle vérifie toutes les contraintes connues.

Pour ne retenir que les possibilités prometteuses, là encore les façons de procéder sont variables :

- évaluation de règles, comme dans le cas de la classification 'heuristique' (Clancey 85).
- évaluation d'une fonction de proximité ou d'intérêt. Par exemple, on privilégie les alternatives les moins lourdes en temps de calcul, ou les plus économiquement prometteuses, si l'on dispose d'un moyen d'en évaluer le coût.
- choix implicite, on prend la première possibilité qui se présente.

4) **Vérification des contraintes**

Une conception sera jugée valable quand elle répondra à toutes les contraintes, qu'elles proviennent des spécifications, ou qu'on les découvre au cours de la démarche.

La vérification des contraintes impose un second cycle d'itérations dans la démarche. Les hypothèses explorées sont ainsi validées ou invalidées au cours de la démarche, sans qu'on puisse dire quand cette étape a lieu.

Une contrainte est une relation qui porte sur (et qui lie) une ou plusieurs données entre elles. Si une contrainte échoue, elle remet en cause les sous-systèmes qui en produisent les éléments, et on est ainsi amené, en remontant la hiérarchie de sous-systèmes concernés par ces éléments, à repartir d'une étape précédente.

Montalban (Montalban 87), fait un long développement pour expliquer que dans un système qui se décompose selon une analyse top-down (des boîtes qui s'emboîtent telles les poupées russes), on passe de spécifications générales à des spécifications de plus en plus fines, quand on conçoit. Partant de contraintes données, on les précise à un niveau de plus grand détail. Ce niveau permet de poser des contraintes à un nouveau sous-système, et on itère la démarche, jusqu'au moment où on ne peut plus décomposer les sous-systèmes obtenus. Cela se comprend, puisque les entrées d'un système sont les entrées des sous-systèmes qu'il contient, et de même pour les sorties.

Pour relâcher une contrainte, et dégager une nouvelle solution, l'important est d'en remonter à la source, que cela soit fait

- automatiquement comme dans Pride (Mittal & 86),
- par l'intermédiaire d'heuristiques, comme dans DSPL (Brown & 89),
- ou encore en rendant la main à l'utilisateur.

Si l'on analyse la façon dont un système peut boucler, là encore deux possibilités restreintes se dégagent :

- les entrées dépendent des sorties (comme dans les cas de recyclage des procédés chimiques). On estime alors une valeur de départ. Bien souvent, c'est l'expert qui est le mieux placé pour la donner, une grande partie de ses compétences réside dans cette appréhension des situations possibles et des ordres de grandeur (cas de PRINCE).

  L'examen des résultats obtenus après la démarche permet de corriger les valeurs de départ, et de relancer le processus, jusqu'à parvenir à la convergence. Cela peut se faire automatiquement comme dans les processus mathématiques itératifs, très communs par exemple dans les problèmes d'optimisation. Ou encore, l'expert sait comment faire pour relancer une nouvelle itération, et ceci peut faire l'objet d'une règle dans le système informatique final. Dans ce cas là, on trouve dans le modèle conceptuel du problème, une tâche appelée 'validation' (cas de DSPL).

- Tout dépend de tout, et on ne sait plus qu'énumérer. Dans cette situation, on se tourne vers les algorithmes à base de contraintes. Ces méthodes vont se trouver particulièrement justifiées, quand le domaine à couvrir présente plusieurs extrema locaux.

IV.2.4.3) POINTS DIFFICILES

Naturellement, toute ou partie de la démarche peut être mise en oeuvre dans un système de conception donné. Par exemple, les règles heuristiques s'appliquent avec bonheur quand il s'agit de faire un choix de matériel, qui est bien une étape de la conception. Si le dimensionnement est déjà fait, ou s'il n'y a pas lieu de le faire, les alternatives ne sont pas affinées.

Des difficultés subsistent dans notre démarche :
• La première difficulté est de **décomposer le problème** en sous-problèmes. Il existe des méthodes d'analyse, non spécifiques des problèmes de conception mais cette décomposition reste un exercice difficile.

L'une des façons de faire, à défaut, c'est de suivre la démarche de l'expert. On prend un exemple, et on le traite avec lui. Puis on restructure ce qu'il a dit et fait.

• Ensuite, il faut **trouver la (ou les) méthode(s) adaptée(s)** à chaque sous-problème identifié. A ce niveau, les traitements ne sont pas forcément ceux que préconise l'expert. Par exemple, manuellement, le travail est bien souvent fastidieux. Aussi l'expert tire une partie de ses compétences de sa faculté de choisir rapidement la bonne méthode, ou les bonnes valeurs. Informatiquement, quand cela n'est pas cher en ressources d'ordinateur, rien n'empêche de tout essayer, ou encore de tenter une première façon de faire et de réitérer en cas d'échec.

• La troisième difficulté est de **faire coexister plusieurs méthodes de résolution distinctes au sein d'une même architecture**. Nous avons vu que chaque sous-problème pouvait faire en soi l'objet d'une démarche. A l'intérieur même d'un sous-problème, plusieurs méthodes de résolution peuvent intervenir. A un certain moment, on pourra utiliser des règles, à d'autres des procédures externes, ou des algorithmes élaborés. De sorte qu'il faut non seulement savoir combiner n'importe quel type de méthodes, mais aussi les agréger ensemble pour retrouver cette notion de sous-problème.

C'est ce point qui a justifié le choix de l'architecture en tâches que nous avons mise en oeuvre dans PRINCE. Il sera détaillé dans le chapitre V *architecture à base de tâches.*

IV.3)EXEMPLES DE SYSTÈMES DE CONCEPTION

Nous essaierons ici de fournir des exemples de systèmes de conception, en se basant d'abord sur les domaines d'application dans le génie chimique. Chacun de nos exemples sera replacé par rapport aux trois classifications, que nous avons introduites.

Puis nous élargirons nos investigations aux systèmes de conception, que nous examinerons sous l'angle des architectures : d'une part, nous introduirons les architectures à base de tâches dont nous nous sommes servis dans PRINCE, d'autre part, nous justifierons l'idée, que nous avons employée précédemment, qu'il n'y a pas de méthode (et partant, pas d'architecture) adaptée à tous les problèmes de conception.

Nous évoquerons ensuite quelques applications marquantes du domaine du génie chimique, toujours dans l'idée de trouver des sources d'inspiration.

IV.3.1) EXEMPLES D'APPLICATIONS

IV.3.1.1) DIMENSIONNEMENT DE PROCÉDÉ, PRINCE

• PRINCE (Wahl 92) est un système de conception de routine, qui a pour charge le dimensionnement des réactions chimiques intervenant dans un procédé, le procédé étant un objet plus abstrait qu'un appareil technologique, ou qu'un process (voir le chapitre II *contexte et problèmes*).

• PRINCE est concerné par des choix de catalyseur, et (surtout) par la fourniture de valeurs numériques, ce qui permet de le ranger dans la catégorie B 'optimisation' (quoique des aspects 'configuration' - F - soient présents). Ce classement n'était pas évident au départ : les ingénieurs parlent au sujet du travail mené par PRINCE, de conception préliminaire.

• PRINCE suit les étapes que nous avons décrites précédemment. On ne génère qu'une alternative à la fois, et chacune est explorée dans son intégralité, avant d'être éventuellement remise en cause par l'expert : l'étape de validation est laissée à la charge des experts, qui décident ou non de reboucler; celui-ci change dans ce cas les conditions appliquées à un catalyseur et relance une simulation. Les étapes de génération des alternatives et choix de ces dernières n'ont donc pas lieu d'être.

Dans le domaine du génie chimique, il n'existe pas à notre connaissance, de système (autre que PRINCE), chargé d'assister les experts responsables des procédés dans leur choix. Les applications existantes sont plutôt centrées sur la mise en oeuvre de modèles cinétiques, chargés de représenter des réactions, ce qui est bien sûr la bonne démarche quand un modèle est disponible, ou s'intéressent à la mise en oeuvre automatique de calcul de génie chimique à

travers des langages spécialisés, ce qui ne peut intervenir que lorsque le schéma de procédé et le niveau des réactions chimiques à faire fonctionner, ont été spécifiés.

IV.3.1.2) CHOIX DE MATÉRIEL

• Les systèmes experts sont naturellement beaucoup employés (Garland 90) (Lesage 90), pour aider au choix de la technique (ou de l'appareil) la (ou le) mieux adapté(e) à un travail donné. Dans ce cas, il s'agit d'abord de **conception de routine**, puisqu'il s'agit d'un choix parmi des possibilités recueillies par ailleurs et ce problème concerne la structure des objets, la fonction de chacun n'étant pas remise en cause.

• Ces applications doivent être rangées dans la catégorie (F) 'Configuration', puisqu'il s'agit bien d'appliquer les contraintes connues, pour faire un choix parmi un certain nombre de structures possibles.

• Ces systèmes exploitent des alternatives connues, recueillies par expertise (il n'y a pas d'étape 'générer des alternatives'), et concentrent leur savoir-faire sur la partie de choix de ces dernières. De telles applications s'arrêtent bien souvent avant l'étape de dimensionnement, et l'étape de validation peut être laissée à la charge des experts.

Un bel exemple en est la famille des Systèmes Experts de Technologie ou SET (Lesage 90), mis au point chez Rhône-Poulenc, pour sélectionner les appareils qu'il faut mettre en place dans les procédés chimiques du groupe, déclinés par classe de matériel. On trouve ainsi un 'set' pour les séparations liquide-solide, un autre pour les pompes, et ainsi de suite. Les règles de sélection sont décomposés en niveau, qui permet un affinement successif des solutions proposées, et la production d'explication. Il est intéressant de noter que la méthodologie des 'set' se reproduit d'une application à l'autre.

IV.3.1.3) FORMULATION

Dans un autre ordre d'idée, SLURRY MINDER (Kelly & 1993) est un système à base de connaissances qui permet de complètement préciser la composition du ciment injecté dans les puits exploités par Schlumberger, notamment de préciser, en fonction du contexte local et du pays d'accueil, les quantités exactes d'additifs qu'il faut ajouter au ciment brut, pour respecter les qualités demandées.

• Dans ce cas, on parle de **conception de routine**. En fait, ce système est très voisin des exemples précédents de choix de matériel.

• Encore une fois ici, on parlera d'**optimisation** (ligne B du tableau), puisqu'il faut trouver les bonnes quantités d'additifs, après avoir **configuré** la formulation (ligne F), en fonction du pays d'accueil.

• Nous trouvons ici les étapes de choix des alternatives et de dimensionnement.

Un autre exemple intéressant est illustré par un système d'aide à la formulation des huiles 'moteur' (Lunn & al 91) Ce prototype a été développé dans le cadre du projet 'Alvey'. Le problème est ici de composer une huile, c'est-à-dire pour l'essentiel choisir les nombreux additifs qui vont être ajoutés, pour obtenir les qualités désirées.

• Pour nous, il s'agit de conception de **routine**, puisque toutes les combinaisons possibles d'additifs ont été décrites.

• La difficulté est de définir le chemin qui va permettre de passer d'une 'base' à un produit fini, chaque étape de ce chemin étant représenté par l'ajout d'un composé, dont il faut de plus définir la quantité. Cette **planification** (ligne D) se révèle d'ailleurs hiérarchique et chaque étape fait l'objet d'une tâche.

• Du point de vue de la démarche, l'étape de génération des alternatives est présente.

IV.3.1.4) SCHÉMAS DE PROCÉDÉS

En conception, il existe une littérature abondante sur la synthèse de schémas de procédé sans que les résultats soient très probants. L'idée de base est de définir l'ensemble des opérations élémentaires nécessaires pour accomplir les transformations souhaitées. Narayanan (Narayanan & 90) propose une architecture inspirée des tâches génériques de Chandrasekaran, pour chaque aspect de la conception d'un 'flowsheet'.

Stephanopoulos (Stephanopoulos 90), faisant le point sur la conception de procédés, déclare qu'il faut construire un système assisté par l'homme ('human aided, machine-based design'), plutôt qu'un système assistant l'homme ('computer-aided design'). Le moins que l'on puisse dire, c'est que l'on en est encore relativement loin. C'est ce qu'avait déjà signalé B. Trousse (Trousse 89) : 'on s'oriente vers des systèmes non pas capables de concevoir, mais guidés par un expert humain'.

Wahnschaft (Wahnschaft 91, 92) présente une application appelée SPLIT, qui s'intéresse aux procédés de séparation tels que les distillations, les 'flash'. Ce logiciel présente une architecture en 'black-board' (Nii 86), qui comprend des sources de connaissances capables de déclencher des procédures du progiciel commercial de génie chimique 'Aspen plus'.

• Ici, il s'agit de conception **innovante**, qui touche aux différentes fonctions des objets qu'il faut assembler : la difficulté est de trouver la bonne séquence de traitement pour parvenir à ses fins.

IV.3.1.5) DIMENSIONNEMENT D'APPAREIL

Les SET de Rhône-Poulenc (voir section IV.3.1.2 : *choix de matériel*) conduisent naturellement vers une démarche de dimensionnement. SPLIT (voir section IV.3.1.4 *schémas de procédé*) incorpore des procédures de dimensionnement à travers sa connexion avec 'Aspen Plus'. En génie chimique, dès qu'on cherche à évaluer de façon plus précise, les paramètres d'une installation, cette étape devient nécessaire.

Un système expert peut être utilisé pour piloter des méthodes numériques dédiés à un problème particulier. C'est ce que montre une application consacrée à un calcul de cyclone (Boldo & 1993).

• De façon évidente, nous sommes dans le domaine de la conception de **routine**.

• Un système expert destiné à faire un choix parmi un certain nombre d'alternatives, se range dans la catégorie F **configuration**. Ceci dit, dès qu'on évoque les aspects de dimensionnement, on mentionne des fonctions d'**optimisation** (B).

IV.3.2) EXEMPLES BASÉS SUR DES TECHNIQUES INFORMATIQUES

Il est en fait difficile de tirer son inspiration du croisement des thèmes de conception et de génie chimique, dans le domaine de l'intelligence artificielle. Aussi, est-il nécessaire d'élargir le champ de nos investigations au domaine, nettement plus large de la conception (prise isolément) et au domaine du génie chimique. Sans nous centrer sur le génie chimique, nous voudrions montrer ici qu'il n'y a pas de technique informatique, plus adaptée qu'une autre au domaine de la conception, ni d'architecture universelle.

Nous nous sommes contentés de trois exemples.

IV.3.2.1) CONTRAINTES

Cette démarche de conception est parfois mise en oeuvre dans les **langages à contraintes,** tel que celui qu'emploie Subramanian (Subramanian 93). Dans ce cas, les étapes de génération des alternatives et de choix deviennent implicites, et ce n'est qu'après avoir posé toutes les contraintes que le système décide de solutions qui les respectent toutes. Cela ne veut pas dire que les étapes de choix, de dimensionnement et de configuration n'existent pas, mais elles sont masquées. C'est un exemple de conception innovante.

Smack (Tsang 91b) utilise également la propagation de contraintes pour combiner entre eux certains appareillages électroniques devant équiper des satellites. A notre connaissance, l'espace de recherches reste de dimension raisonnable, le

nombre de blocs électroniques de base étant assez réduit (de l'ordre de la dizaine). Au vu de la complexité, cette application pourrait être rangée dans la catégorie conception de routine : il est possible de recenser toutes les situations.

En fait les langages à base de contraintes présupposent que toutes les contraintes soient posées avant que le problème puisse être résolu. Malheureusement, comme le remarque S. Mittal (Mittal 90), dans les problèmes de conception, les contraintes sont bien souvent dynamiques, c'est-à-dire que ce n'est qu'au cours de la résolution qu'elles sont précisées, au fur et à mesure qu'une solution se dégage. Ce problème commence tout juste à être abordé (Berlandier & 93).

IV.3.2.2) CASE-BASED REASONING

C'est dans les raccourcis qu'un expert n'hésite pas à emprunter, que toute la difficulté de la démarche de conception réside. Un ingénieur, chargé de proposer des solutions, fera des impasses suivant le problème qu'il traite, affinera une hypothèse plutôt qu'une autre, se servira de son expérience passée, sautera une étape de dimensionnement parce qu'il en connaît le résultat.

Il est cependant tentant pour adapter une solution à un nouveau problème de repartir de cas que l'on a déjà traité. En cela, les méthodes de '**case-based reasoning**' peuvent trouver là un champ d'application important. Un exemple en est le logiciel prototype 'kritik' (Goel & 89) -dont le nom signifie conception en sanskrit (sic)- ou encore 'DESIGN' (Hall 90), où les éléments du problème (les pièces) sont recherchés dans deux bases de données. Comme un composant peut se décomposer de plusieurs façons, que les dénominations sont variables, cette recherche est loin d'être facile.

Le logiciel CLAVIER, cité dans un livre de J. Kolodner (Kolodner 93) est utilisé chez le fabricant d'avions Lockheed et sert à disposer un ensemble de pièces en matériau composite dans un four, pour qu'elles soient toutes traitées de façon efficace. C'est avant tout un problème de configuration, qui s'est inspiré au départ d'un petit nombre d'exemples de base.

Le raisonnement à base de cas convient bien à la conception de routine : on tire, de la base de cas, un (ou plusieurs) exemple(s) le(s) plus proche(s) possible(s) et on l'adapte (ou les adapte). Les exemples préexistent, il ne s'agit pas de les rebâtir, ils constituent une première approximation du cas à traiter.

IV.3.2.3) TÂCHES

Décomposer un problème, c'est dégager une architecture à base de tâches, qui trouvent là un emploi naturel. PRIDE (Mittal & 86a et b) prônait déjà une telle mise en forme, reprise, avec des différences, dans ARCHIX (Thoraval & 90) par exemple.

Utiliser des tâches suppose que le problème ait été préalablement analysé dans tous ses détails. On tombe donc plutôt sur des problèmes de **conception de routine**, sans exclusive.

L'enchaînement des tâches peut être fixé au moment de la réalisation, comme dans DSPL (Brown & 89) , ou variable à l'exemple de LISA (Delouis 93) et redéterminé à chaque passage. SMECI (Smeci 92) est un générateur de système expert, dont l'utilisation par défaut suppose cet enchaînement fixe.

Le raisonnement en tâches s'applique bien aux problèmes de conception : on décompose, souvent, ces problèmes par étapes, de façon hiérarchique, ou séquentielle, comme l'a fait remarquer H.A. Simon (voir le début de ce chapitre) et les tâches s'emploient naturellement au moment de l'analyse et du recueil de l'expertise. Comme ensuite, ces mêmes tâches sont maniées par les experts quand ils utilisent le système, les tâches offrent naturellement un cadre de représentation qui permet aux utilisateurs (experts), comme aux réalisateurs (cogniticiens) de parler le même langage.

D'autres raisons militent pour développer une telle architecture (en particulier, la facilité d'utilisation des tâches), que nous ne détaillerons pas ici (voir le chapitre V *architecture à base de tâches*).

IV.3.3) EXEMPLES DU DOMAINE DU GÉNIE CHIMIQUE

Certaines applications présentent tellement de facettes, qu'il est difficile de les ranger dans une catégorie ou une autre, et c'est pourquoi il est utile de dépasser la cadre strict de la conception.

Le domaine du pétrole est particulièrement vaste. Quand il est exploité en raffinerie, les techniques mises en oeuvre sont issues du **génie des procédés**. Le domaine du pétrole n'amène pas de ce point de vue de spécificité particulière.

P. Bourseau (Bourseau 93) fait une synthèse des applications de l'intelligence artificielle dans le domaine du génie des procédés. D'après lui, les applications les plus nombreuses concernent la supervision des procédés, c'est-à-dire le diagnostic de pannes, le traitement d'alarmes, l'aide à la conduite, et la maintenance préventive. DIAPASON, pour DIAgnostic Prédiction Avec SimulatiON (Penalva 91) est un système qui tente d'intégrer bon nombre de ces aspects

B. Braunschweig (Braunschweig 90) passe en revue les réalisations de l'intelligence artificielle dans le domaine du pétrole (voir aussi AFIA 94). Le croisement de ces deux champs d'étude (IA et pétrole) couvre encore un très large éventail, qui va du forage jusqu'au procédé. Certaines applications se signalent par leur taille, comme ODDA (Cayeux 92), qui concerne la technique du forage. Des réalisations plus modestes ont conduit à des gains économiques considérables : ainsi le suivi de l'utilisation du gaz dans la raffinerie de Grand Puits (Hartmann 93), ou encore le système expert de conduite "on line" d'une

unité de désalphaltage (Boucot & 91) : les tonnages traités sont tels que des gains même marginaux par tonne traitée, dégagent sur une année, des sommes conséquentes.

IV.3.3.1) ENVIRONNEMENT INTÉGRÉ

Il faut bien sûr citer DESIGN-KIT (Sriram & 89), conçu pour offrir un ensemble d'outils qui accompagnent le développement des procédés. DESIGN-KIT permet la saisie interactive des schémas de procédé, la génération automatique des équations de bilan, le lancement de session de simulation numérique et le raisonnement sur les ordres de grandeur. A l'intérieur de DESIGN-KIT, MODEL_LA est un langage dédié, destiné à modéliser les schémas de procédé.

Le système SPI, conçu par Bourseau et Mizessyn (Bourseau 93) et appliqué en particulier aux fours à ciment est aussi un ensemble intégré d'outils. L'idée de base est d'assister l'ingénieur dans la saisie du schéma de procédé, grâce à une interface graphique intelligente, en émettant des diagnostics et en complétant automatiquement certaines informations dès que deux appareils sont connectés. Par exemple, *"sur une installation cimentière, un flux allant du four au refroidisseur contient de l'air et son nom est "air secondaire"* (p 159). Une liaison entre un four et un refroidisseur est donc nécessairement un flux d'air secondaire, l'utilisateur n'a même pas besoin de le préciser.

IV.3.3.2) ENVIRONNEMENT DÉDIÉ

Dans cette rubrique, on peut citer ODDA (Cayeux 92), qui, en tant qu'environnement dédié, rassemble des techniques d'aide au forage, depuis le choix de l'emplacement des têtes de puits, ce qui est typiquement un acte de conception, jusqu'au suivi du forage, qui s'inscrit dans la rubrique 'pilotage', et relève de la notion générique de diagnostic. Le choix du trajet que doit suivre la colonne de forage est basé sur l'application de gros codes de calculs, et ODDA offre pour cela l'environnement d'utilisation de ces programmes. Après bien des étapes, ODDA fournit une aide au suivi du forage, et c'est là semble-t-il son atout principal : il s'agit d'éviter que la colonne de forage ne casse, les pertes économiques pouvant être conséquentes quand c'est le cas. ODDA agit là comme un système d'aide à la conduite, en conseillant les experts foreurs. En pratique, le forage d'un puits dure plusieurs semaines ou mois, et les techniciens qui s'occupent de forer ne sont pas forcément les mêmes d'un bout à l'autre de l'opération. ODDA se comporte comme un réceptacle de l'expertise, et permet de transmettre un certain savoir-faire d'un ingénieur à l'autre.

IV.4) CONCLUSION

A travers les différentes revues que nous avons effectuées, nous avons dégagé qu'un système de conception touche la structure ou la fonction des objets à concevoir. Une conception de routine permet essentiellement d'adapter la structure des objets, tandis que l'innovation s'appuie sur un ensemble de fonctions élémentaires dont la combinaison est novatrice.

De notre point de vue, un système de conception suit la démarche suivante : il commence par analyser les spécifications qui lui sont fournies. S'appuyant sur une décomposition en sous-problèmes effectuée par ailleurs, il génère un certain nombre d'alternatives par sous-problèmes. Il retient les possibilités valides, qu'il affine en les dimensionnant, puis il applique les contraintes connues pour passer au sous-problème suivant ou au contraire réitérer.

PRINCE est un système de conception de routine. Il fait du dimensionnement (c'est-à-dire qu'il se range dans la catégorie configuration). Il suit la démarche précédente, quoique de façon simplifiée.

Il est intéressant de noter que, dans le domaine du génie chimique (pour le moins), nous n'avons pu trouver aucun logiciel dont les fonctionnalités soient proches de PRINCE, c'est-à-dire un logiciel qui met en oeuvre une architecture capable d'intervenir dans la prédiction des réactions chimiques, couplée à des visualisations graphiques. C'est ce qui explique que nous allons présenter dans les deux chapitres suivants, d'une part l'ARCHITECTURE À BASE DE TÂCHES, pour la partie prédiction, et d'autre part, toute la réflexion portée sur les COURBES et les visualisations.

CHAPITRE V

ARCHITECTURE À BASE DE TÂCHES

V.1) LA DÉMARCHE DE RÉSOLUTION

L'observation des experts, du point de vue de la modélisation des connaissances et l'analyse des problèmes qu'ils traitent, du point de vue de l'interaction homme-machine, nous ont convaincus qu'il fallait mettre en oeuvre une architecture à base de tâches. Aussi la première partie de ce chapitre rapportera la façon dont les experts s'y prennent pour résoudre les problèmes qui leur sont soumis. Il ne s'agira pas, pour nous, de décrire les différentes étapes de la démarche, mais plutôt de retranscrire l'utilisation des différents outils qui sont à leur disposition, quand ils répondent à un appel d'offre. Nous justifierons ainsi le choix de l'architecture adopté pour PRINCE, ce qui conclura cette première partie.

Dans la seconde partie, nous détaillerons la façon dont nous avons implanté cette architecture à base de tâches dans PRINCE. Si l'on prend un équivalent avec les systèmes experts de première génération, la base de faits est donnée à chaque fois par le cas d'étude traité (voir le chapitre III intitulé *un cas d'étude*), **la base de règles** est représentée par les tâches et **le moteur d'inférences** permet la résolution. Notre présentation suivra ce découpage en règles et moteur.

Nous commencerons par analyser les raisonnements de façon statique (section V.2.2). Pour simplifier, les raisonnements dans PRINCE sont représentés par des opérateurs de nature quelconque (formules, règles, programmes,...), liés par des données. Nous montrerons comment nous avons modélisé ces notions sous forme d'objets (**objets relations et flux**), en tenant compte à chaque fois des **possibilités d'inversion** de ces opérateurs.

Nous continuerons en expliquant comment nous avons exploité ces connaissances dynamiquement (section V.2.3), pendant la résolution.

Cependant, la notion de tâche est complexe. Aussi, avant de parler des tâches, nous définirons au préalable les concepts principaux que nous emploierons (notamment ceux de flux et de relation), après avoir rappelé succinctement notre interprétation des mots tâche et but (section V.2.1).

Enfin, en troisième partie, nous replacerons ce type d'architecture dans son contexte bibliographique, c'est-à-dire que nous essaierons de comparer notre utilisation de ce concept avec ce que d'autres en ont fait. Si le lecteur le souhaite, il peut lire en premier cette troisième partie.

V.1.1) OBSERVATION DES EXPERTS, LORS DE LA RÉSOLUTION D'UN PROBLÈME

Quand on interroge les experts sur leur façon de faire pour concevoir un réacteur de raffinage, ils décrivent un processus relativement séquentiel, et dégagent à son propos une trentaine de tâches successives faiblement couplées entre elles. Ceci ne signifie pas que les retours arrière sont interdits, mais que l'ordre des tâches est prédéterminé. Par exemple, on n'attaquera pas la détermination de la composition des produits avant d'avoir précisé la température de réaction même si plusieurs essais sont nécessaires pour cela. Chaque tâche se décompose à son tour, pour former finalement un arbre de tâches et de sous-tâches. Cette décomposition en tâches et sous-tâches est une démarche communément appliquée en Intelligence Artificielle, dans le domaine de la conception notamment (Mittal & 86), (Myers & 88), (Neveu & 90).

L'observation précise des experts au moment de la résolution du problème de conception ne confirme pas complètement cette manière de voir. Il y a un certain écart entre ce discours, construit dans une situation où les experts sont poussés à rationaliser leur démarche, et leur façon concrète et pratique d'aborder le problème.

V.1.1.1) ILS NE PROCÈDENT PAS DE MANIÈRE SÉQUENTIELLE

Comme le remarque W. Visser, spécialiste de psychologie cognitive, plutôt que d'attaquer les problèmes séquentiellement, les experts procèdent de manière globalement opportuniste (Visser 91), c'est-à-dire qu'ils sautent d'un sous-problème à un autre de manière apparemment désordonnée, en fonction des informations disponibles et de la difficulté des sous-problèmes qu'ils traitent. Par exemple, pour se faire une idée plus précise d'une question qui demande de longs efforts, on s'attaque à quelques points isolés qui donnent une première estimation de la réponse possible.

Notre idée dans le système à base de connaissances PRINCE est d'apporter une assistance aux experts, plutôt que de leur imposer une démarche. C'est pourquoi nous avons voulu leur donner cette possibilité d'utiliser l**es tâches de façon isolée,** sans lien avec le reste de la démarche. Si on choisit une tâche, qu'on lui fournit ses valeurs d'entrée, après lancement, elle calcule sa (ou ses) valeur(s) de sortie.

V.1.1.2) LES TÂCHES DÉGAGÉES SONT DE NATURE QUELCONQUE.

Toutes les tâches utilisées par les experts ne sont pas de la même espèce : *par exemple, pour calculer le taux de la réaction de désazotation (appelé HDN), les experts vont exploiter un réseau de points expérimentaux entre lesquels ils interpolent. Une fois qu'on a obtenu l'HDN, le niveau de l'azote dans l'effluent se calcule par une formule.* Parfois, il est nécessaire d'écrire un petit programme pour déterminer le résultat, comme dans le cas du *calcul de la densité de*

l'effluent à l'issue du craquage. Certaines valeurs sont carrément l'objet de règles expertes. Et ainsi de suite.

De telle sorte que pour mener à bien l'application, il faut savoir **manipuler n'importe quel type de tâche**, qu'il s'agisse de règles, ou de calculs plus ou moins complexes. L'architecture à base de tâches que nous avons développée est non seulement capable de faire fonctionner les types de tâches que nous avons énumérés, mais aussi capable d'en intégrer de nouvelles, au prix de manipulations informatiques réduites.

V.1.1.3) LES TÂCHES SONT PARFOIS INVERSIBLES

Un opérateur chevronné pour conduire une unité, ou un concepteur expert pour la vendre, manie bien souvent des raisonnements inverses : Par exemple, *plus la charge est difficile, plus il est nécessaire de monter la température de réaction. Inversement, plus la température de réaction augmente plus la réaction est efficace*. Rappelons que les unités dans l'industrie du raffinage peuvent être énormes, les tonnages dégagés conséquents, il ne s'agit donc pas d'un jeu.

Cette possibilité d'inversion est loin d'être négligeable. En fait, on peut considérer que c'est grâce à ce mécanisme que les experts peuvent raisonner et obtenir des certitudes, dans le réseau d'inférences et d'abaques qui est leur domaine.

Cette caractéristique d'inversion a été systématiquement mise en oeuvre dans PRINCE, et largement exploitée.

V.1.1.4) ILS PARTENT DES OBJECTIFS

Le problème de dimensionnement de procédé chimique consiste à fixer les conditions opératoires, c'est-à-dire les conditions d'utilisation du procédé, à des valeurs qui permettent d'atteindre, le mieux possible, les objectifs fixés par le demandeur (le client qui émet l'appel d'offre dans notre cas). Comme les variables intervenant dans un procédé chimique sont continues (une pression, une température peuvent prendre n'importe quelle valeur dans un domaine déterminé), il est impossible d'énumérer toutes les solutions possibles, qui sont en nombre infini. Dès lors, pour déterminer les conditions opératoires, on part des objectifs et on parcourt en arrière l'enchaînement hiérarchique des causalités décrites par les experts.

Une démarche purement séquentielle ne nous permettrait pas d'obtenir toute la souplesse nécessaire. Une résolution par **chaînage arrière**, qui part des objectifs, permet de tenir compte des données disponibles et uniquement de celles-là, et une étape de raisonnement n'est lancée que si toutes les informations nécessaires sont présentes. Le système n'ira pas chercher des données manquantes qui n'ont rien à voir avec le problème posé. Le chaînage arrière permet de poser les bonnes

questions. (Nous verrons cependant, en annexe, qu'il existe d'autres solutions que le chaînage arrière pour traiter cette question.)

V.1.1.5) LES DONNÉES DIFFÈRENT D'UN CAS À UN AUTRE

Les spécifications que les clients émettent ne sont jamais les mêmes d'un appel d'offre à l'autre. En effet, les objectifs diffèrent en nature et en niveau. Par exemple, un client privilégiera le *bas niveau du taux de soufre dans l'effluent*, tandis qu'un autre sera plus attentif à obtenir *le maximum d'essence à partir du pétrole brut*. Ces différents desiderata conduisent, de fait, à des différences dans les étapes de la résolution. Par voie de conséquence, il n'y a pas qu'un problème de conception, mais pratiquement **autant de problèmes que de types de résultats à obtenir.**

Les données fournies par le client diffèrent entre deux appels d'offre, changent d'une variante à une autre. Les charges pétrolières sont décrites différemment d'un appel d'offre à l'autre et les demandes concernent des schémas d'unité plus ou moins bien définis, puisqu'il peut s'agir d'unité neuve, de remplacement de catalyseur (avec 'revamping' -i.e., modification du schéma- ou non).

Bien évidemment, les experts et les ingénieurs chargés de répondre à ces appels d'offre savent reconstruire une démarche à chaque fois, c'est-à-dire se fixer une suite de buts successifs qui leur permet de lever toutes les indéterminations, et trouver les valeurs inconnues.

C'est pourquoi, outre cette possibilité que nous avons évoquée dans le paragraphe précédent de reconstruire une démarche automatiquement par chaînage arrière, nous avons introduit la faculté pour les experts et les ingénieurs utilisateurs de PRINCE, de fixer et d'introduire eux-mêmes la suite de buts qu'ils souhaitent que le système atteigne. Les utilisateurs ont la possibilité d'**enchaîner des tâches.**

V.1.1.6) CONCLUSION

Pour se rapprocher de la façon de faire des experts, on dégage des sources de connaissance appelées tâches. Ces tâches sont de nature quelconque. On peut les utiliser indépendamment des autres; ou on peut les enchaîner pour construire des raisonnements en partant des objectifs, que cela soit fait automatiquement, ou suivant une suite de buts préétablis. Enfin , les tâches se manipulent dans le sens direct ou inverse.

V.1.2) JUSTIFICATION D'UNE ARCHITECTURE À BASE DE TÂCHES

Nous venons donc d'observer que la démarche des experts est dirigée par les buts, c'est-à-dire qu'elle s'organise en fonction des résultats qu'ils souhaitent atteindre. Une construction informatique classique imposerait de recenser toutes les situations possibles, pour être sûr de collecter les informations nécessaires à chaque fois. Il s'agit pour nous de renverser la perspective, pour donner aux experts la possibilité de **raisonner avec le système informatique comme ils ont l'habitude de le faire à la main.**

Précisons que notre objectif n'est pas de reproduire le cheminement précis des experts, tel qu'un observateur, un psychologue par exemple, aurait pu le noter, en dépouillant en présence de ceux-ci, plusieurs cas d'appels d'offre. Ce cheminement dépend des cas traités et de l'expérience des utilisateurs. La formalisation de l'expertise serait dans ce cas particulièrement ardue et nous ne serions pas sûrs de traiter tous les cas, ni d'arriver systématiquement aux bonnes solutions.

Une façon à la fois plus élégante et plus simple de contourner cette difficulté, est de mettre directement les **buts** mis en évidence dans l'analyse de la manière de faire des experts **à disposition des utilisateurs.** Ils auront ainsi toute latitude pour conduire leur démarche eux-mêmes.

L'architecture informatique découle des propositions que nous venons de formuler, elle est basée sur les **tâches** (voir la partie suivante, pour une définition plus précise de ce mot). Cette architecture présente pour nous les avantages suivants :

• 1) Opérationnalisation
Si l'on admet que ce mode de raisonnement se retranscrit sans déformation majeure dans le schéma conceptuel informatique, on peut parler comme I. Delouis (Delouis 93), d'opérationnalisation du schéma conceptuel, c'est-à-dire de la possibilité donnée aux utilisateurs de directement manier la représentation qui est la plus proche de leur façon habituelle de procéder. Comme chaque tâche porte le but qu'elle est censée atteindre, une architecture à base de tâches s'inscrit naturellement dans une interface déclarative, plus proche de la façon de penser des experts.

• 2) compétence
Les connaissances exploitées dans l'application sont, on l'a vu, souvent corrélatives. Or, une corrélation n'a cours que dans un domaine précis, qu'on n'extrapole pas sans risque. Une architecture à base de tâches, comme le souligne David (David & 90), permet de cerner les limites de compétence du système, au niveau de granularité souhaité par les experts.

• 3) coopération homme-machine
Les tâches sont utilisées pour attribuer à l'utilisateur la résolution de problèmes que le système informatique ne peut résoudre de façon autonome (Stolze 91) (Delouis 93) (Willamowski 94a).

4) modularité
Notre manière de mettre en oeuvre la coopération est légèrement différente : l'utilisateur peut manipuler les tâches comme bon lui semble, en en lançant une de façon isolée, quand il veut un résultat ponctuel, en construisant des suites de buts qu'il veut que le système accomplisse, c'est-à-dire en enchaînant les tâches, ou en laissant le système se débrouiller automatiquement. L'utilisateur garde ainsi très souplement le contrôle du système, en ne lui laissant faire que ce qu'il souhaite.

• 5) explication
Un découpage en tâches permet, c'est le dernier avantage que nous voudrions signaler, d'expliquer au bon niveau de détail, c'est-à-dire au niveau de détail souhaité par les experts, le raisonnement suivi par le système (Steel 90) (Chandrasekaran & 92). Cette possibilité d'explication n'a pas en fait encore été exploitée dans PRINCE, mais elle fait l'objet de demandes de la part des utilisateurs.

L'organisation de l'application PRINCE en tâches s'inspire d'une longue tradition en Intelligence Artificielle (cf section V.3). PRIDE (Mittal & 86a et b) utilisait déjà les concepts de buts, sous-buts et tâches. D. Brown (Brown & 89) a créé DSPL (Design Structures and Plans Language), un langage destiné aux conceptions de routine. Dans DSPL, on note l'utilisation de plans, de tâches et d'items. Plus près de nous, on peut citer TASK (Pierret-Golbreich 90), qui n'a pas eu à notre connaissance d'application industrielle, mais qui a inspiré LISA (Delouis 93), qui s'applique aux problèmes de réseaux électriques d'EDF. B. Rousseau (Rousseau 88) a monté une architecture basée sur les tâches, pour créer un environnement de résolution de problèmes, domaine qui suscite encore de nouvelles réflexions (Chaillot & 94) (Willamowski 94b).

V.2) TÂCHES DANS PRINCE : DÉFINITIONS ET MISE EN OEUVRE

Les considérations développées dans la section précédente conduisent à mettre à la disposition des utilisateurs toutes les tâches mises en évidence dans le modèle conceptuel, parce que ce sont celles que les ingénieurs manipulent. Avant de commencer à décrire la mise en oeuvre des tâches dans PRINCE, nous allons introduire les définitions indispensables.

Nous débuterons en précisant ce que nous entendons par le mot tâche, puis nous poserons les quelques définitions nécessaires. Ces définitions font parfois référence à des notions que nous développerons plus loin dans la partie bibliographique. **Les deux notions principales sont celles de relation et de flux.**

Pour exploiter la notion de tâche dans PRINCE, nous avons défini une architecture capable de les représenter, puis mis en oeuvre une méthode de résolution pour les exploiter. La présentation de l'application suivra ce découpage.

V.2.1) DÉFINITIONS

V.2.1.1) SENS DES MOTS TÂCHE ET BUT

Le dictionnaire 'Le Petit Robert' précise qu'une tâche est un 'travail déterminé qu'on doit exécuter', tandis qu'un travail est 'l'activité nécessaire à l'accomplissement d'une tâche'. Au-delà de la référence croisée (tâche renvoie à travail et inversement), il est intéressant de noter la mention de l'activité, plutôt que de l'objectif à atteindre, dans ces définitions.

A une tâche correspond un but et il est important, pour éviter les confusions de distinguer buts et tâches. Pour notre part, nous allons postuler que :

1) **un but désigne toujours l'obtention d'un résultat à l'issue d'une activité.** Par exemple, *'generate hypotheses'* (voir schéma n°17 des tâches de KADS, section V.3.2.1) signifie obtenir certaines hypothèses, après exercice de l'activité de production de ces dernières. Dans PRINCE, le but *'caractériser la charge '* peut s'exprimer comme définir une charge *'caractérisée '* (la sortie), après avoir appliqué l'action *'caractériser '* sur une charge *'non qualifiée '*, en entrée.

2) **Une tâche désigne l'application d'une action que l'on peut plus ou moins préciser, en vue de parvenir à un certain objectif,** plutôt que le but à atteindre.

Chaque action suppose un modèle de l'activité à entreprendre, modèle qu'on peut préciser en décrivant la méthode que l'on va employer.

Cette notion de tâche est en fait difficile. Quand on parle d'une tâche comme 'couper du bois', l'activité évoquée est une abstraction de l'activité réelle qui sera mise en oeuvre, qui pourrait être : 'couper du bois avec la scie électrique pendue dans l'appentis, à partir de troncs d'arbres rassemblés à cet effet dans le hangar' par exemple. Comme le dit Vogel (Vogel 88 p 40), 'une action ne trouve pas sa source dans un objet réel, mais dans un projet abstrait'. Une des difficultés de la notion de tâche est qu'on référence une action concrète et précise en en parlant de façon abstraite et générale.

Cette ambiguïté est bien prise en compte dans (Chaillot 93). Quand on évoque une tâche (en général), on fait référence à l'ensemble des méthodes susceptibles d'atteindre un but donné, mais quand on mentionne une tâche particulière, on désigne une méthode donnée.

Pour résumer, un utilisateur, quand on lui parlera de tâche, pensera à un modèle d'action à entreprendre en vue d'atteindre un objectif, dans un contexte particulier, plus ou moins précisé, plutôt qu'à la désignation d'un objectif lui-même.

conséquences

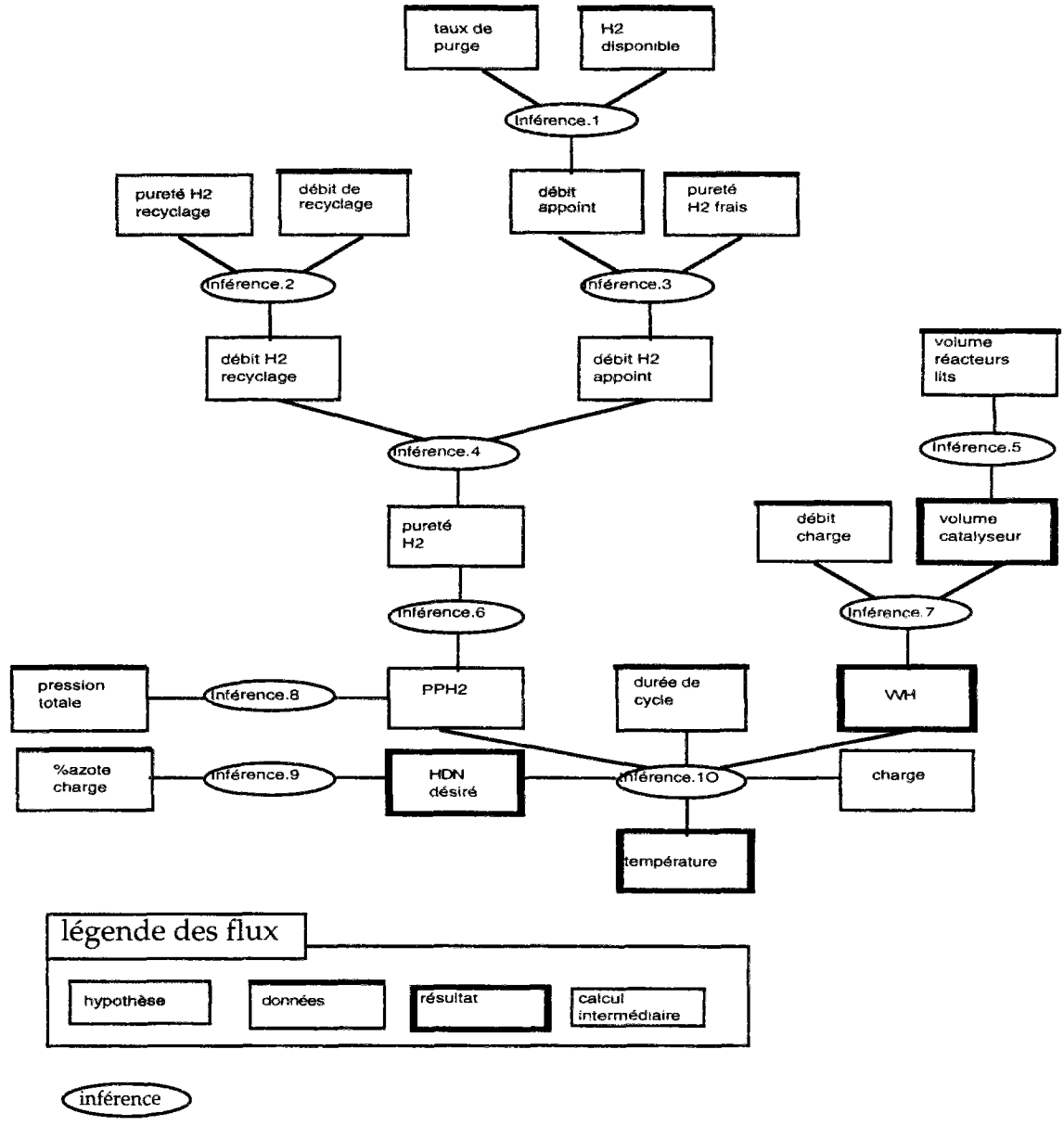
Si les tâches font référence à des activités, il est juste de parler de tâche quand on examine le fonctionnement d'une application. Le mot **tâche est employé pour décrire la dynamique de résolution. On évoque des inférences (des pas de raisonnement) en statique**. Le vocabulaire n'est pas le même, alors que les notions sous-jacentes sont très voisines.

V.2.1.2) REPRÉSENTATION STATIQUE DES CONNAISSANCES

Les notions les plus importantes de notre point de vue sont celles de **relation et de flux** : un raisonnement est vu comme un ensemble d'opérateurs (qui donnent des relations) reliés par des flèches (ce qui se traduit par des flux). Les concepts de relation et de flux permettent de déduire ceux d'inférence, et de but notamment.

Pour représenter le raisonnement suivi par les experts, il est commode de tracer un Diagramme de Flux de Données (ou DFD), comme dans le schéma n° 1 ci-dessous. Un pas de raisonnement s'appelle une inférence. Cette représentation est sous-jacente à toutes nos définitions.

schéma 1: **diagramme de flux de données, pour la détermination de la température de réaction**

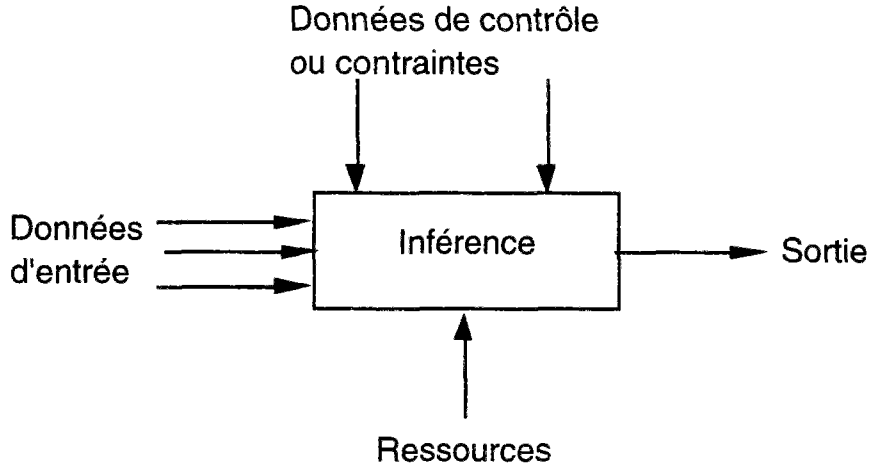


Nous introduisons maintenant les notions d'inférence, de relation, de flux et de but pour représenter statiquement le raisonnement. L'exemple du paragraphe V.2.4.1 illustre toutes ces définitions.

Définition d'une **inférence** : une inférence est un pas de raisonnement. Il peut être élémentaire ou non, c'est-à-dire qu'il peut éventuellement être décomposé.

Cette inférence peut se représenter comme un actigramme (au sens de SADT), c'est-à-dire d'une boîte (représentant quelque chose que l'on fait), liant des données d'entrée à des données de sortie, sous certaines conditions ou contraintes.

Schéma 2: **représentation d'une inférence à la façon des actigrammes de SADT.**



Les ressources (fichiers, ...) correspondent au support dans SADT, les contraintes aux données de contrôle, les inférences aux activités.

Les inférences disposent des ressources (par exemple *un fichier de points de définition quand il s'agit d'interpolation dans un réseau de courbes*) qui leur sont nécessaires, connaissent les conditions (les contraintes) qui autorisent leur activation, savent identifier leurs entrées, et leur sortie. Elles permettent de faire un pas de raisonnement, qu'il soit élémentaire ou non, c'est-à-dire qu'il se décompose ou non.

Exemple : *la formule de calcul du taux de désazotation à partir de la quantité d'azote de la charge, et de la quantité d'azote souhaitée dans l'effluent, permet de définir une inférence. Les entrées sont la quantité d'azote (A_c A pour Azote) de la charge, et la quantité d'azote souhaitée dans l'effluent (A_e). La sortie est le taux de désazotation HDN, ce qui signifie 'hydrodenitrogenation', ou hydrodésazotation, HDN étant l'abréviation utilisée par les experts. La méthode employée est la résolution d'une équation : $HDN = (A_c - A_e) / A_c$.*

Nous allons imposer maintenant deux restrictions aux inférences :

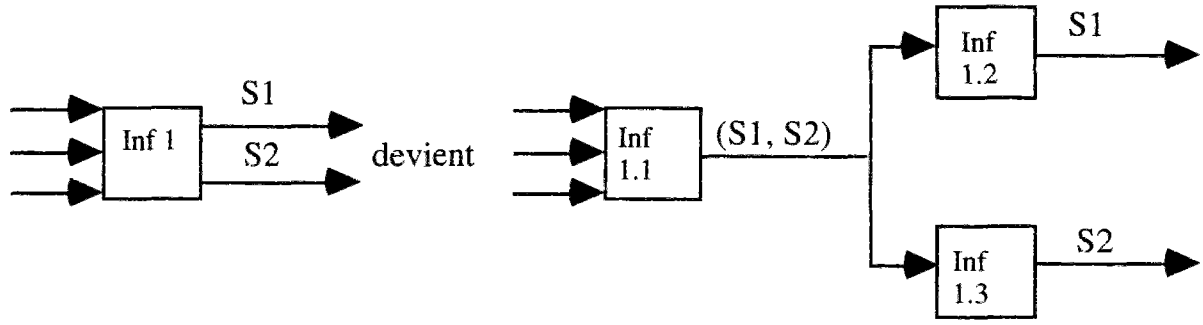
Propriété sortie d'une inférence : Une inférence a toujours une et une seule sortie.

Cette restriction est imposée pour éviter des problèmes de cohérence avec les buts. Nous expliquerons ce point quand nous examinerons les liens entre les tâches et les problèmes.

Une inférence qui ne rend aucun résultat n'existe pas. Ce résultat peut être vide.

Quel que soit le nombre a priori de sorties de l'inférence, il est facile de se ramener à un schéma où les inférences considérées n'ont plus qu'une seule sortie, comme l'exemple ci-dessous l'illustre.

schéma 3 : **décomposition d'une inférence à 2 sorties**



Les inférences 1.2 et 1.3 sont des opérateurs extrêmement simples de projection.

Propriété fonction sous-jacente à une inférence : La transformation sous-jacente à une inférence est une fonction, c'est-à-dire qu'à un ensemble donné de valeurs d'entrée, il ne peut correspondre qu'une valeur de la sortie.

Définition d'une **relation** : une inférence utilise une **relation** entre des données d'entrée et de sortie.

Une relation, contrairement à une inférence, peut être inversible, par rapport à une certaine donnée. Si, par exemple, une relation lie les données d'entrée D_{e1} et D_{e2} à une donnée de sortie D_s, et si cette relation est inversible par rapport aux données d'entrée D_{e1} et D_{e2}, elle permet de définir trois inférences : celle qui fait passer de D_{e1} et D_{e2} à D_s, celle qui va de D_{e1} et D_s à D_{e2}, celle qui va de D_{e2} et D_s à D_{e1}. Notons qu'une relation peut ne pas être inversible par rapport à toutes les données d'entrée.

Nous utilisons plutôt cette notion de relation que celle d'inférence. D'un point de vue informatique, elle est plus synthétique : une relation permet de déduire n inférences, autant que de sortie possibles.

Définition d'une **méthode** : pour s'exécuter, une inférence fait appel à une **méthode** de résolution.

En fait, une inférence entre des entrées et une sortie renvoie à une méthode de résolution, et réciproquement. Une inférence, quand elle est recueillie auprès des experts, n'existe que parce qu'il y a une méthode qui permet de transformer les entrées en sortie. C'est pourquoi, comme nous le constaterons en étudiant d'autres auteurs, méthode et inférence sont des mots souvent employés l'un pour l'autre.

Comme les inférences se déduisent des relations, on mentionne improprement, par abus de langage, la méthode sous-jacente à une relation. Il y a en fait deux méthodes liées à une relation : la méthode directe et la méthode inverse, si elle existe.

Une méthode est de nature quelconque : il peut s'agir de formule comme dans le cas précédent, mais on trouve également des règles, des corrélations 'graphiques', qui se traduisent par des courbes définies par des points, des programmes.

Une méthode peut être élémentaire, comme quand elle s'appuie sur une formule de calcul, une procédure algorithmique, un ensemble de règles. Parfois, elle est décrite en plusieurs étapes (chacune d'entre elles pouvant être un but) : elle est alors composée. Enfin, comme dans KADS, une méthode peut être de transfert, comme lors d'une interaction avec l'utilisateur.

Une inférence, comme la méthode qu'elle utilise, sera élémentaire, composée, ou de transfert.

Exemple : une **méthode élémentaire** (quoique sophistiquée) est la méthode d'*interpolation dans des abaques* . Cette méthode est utilisée de nombreuses fois dans PRINCE, bon nombre de pas de raisonnement étant basés sur l'exploitation de courbes. Pour citer un cas, obtenir le taux de désazotation, à partir de la connaissance de la charge et des conditions opératoires se fait de cette manière.

La *caractérisation de la charge* repose sur une **méthode composée**. Elle s'effectue en deux étapes, d'abord une vérification des valeurs de la charge, pour savoir si aucun chiffre n'est aberrant, puis une évaluation de la difficulté du traitement à venir. Ces deux étapes sont basées, dans ce cas, sur des règles expertes.

Définition d'un **flux** : un flux est une entrée ou une sortie d'une inférence.

Exemple : *les entrées (A_c, A_e) et la sortie (HDN) de l'inférence qui définit HDN comme le résultat de la formule (A_c- A_e)/A_c (voir la relation précédente) sont des flux*

Ces flux peuvent être ou non en correspondance avec le domaine, avoir ou non une signification pour l'utilisateur.

Proposition **caractérisation des flux** : un flux peut être un flux d'entrée, quand il est exclusivement utilisé en entrée d'une relation, un flux de sortie (quand il résulte de l'application des flux d'entrée à la relation, sans jamais pouvoir être utilisé en entrée), ou d'entrée-sortie.

Exemple : *les flux A_c, A_e, et HDN de la relation précédente sont du type entrée-sortie., puisque la formule permet également de calculer A_c à partir de A_e et de HDN, et A_e à partir de A_c et de HDN.*

Proposition **utilisation des flux** : pour s'exécuter une relation nécessite :
- la fourniture d'une valeur à tous les flux d'entrée
- l'alimentation de tous les flux d'entrée-sortie sauf un.

Cette proposition signifie que si toutes les valeurs d'entrée sont connues, la relation va pouvoir s'exécuter, c'est-à-dire vérifier ses conditions de lancement et si ces dernières sont satisfaites, traiter les entrées pour obtenir le résultat.

Définition d'un **but** : un but est un flux de sortie d'une inférence, visible par l'utilisateur.

Cette définition renvoie au paragraphe V.2.2.1. Un but qualifie l'obtention d'un résultat (à l'issue d'inférences), et l'aboutissement d'un raisonnement (comme dans KADS).

Au cours du recueil de l'expertise et de la construction des raisonnements, un but n'est posé que parce que l'on dispose d'un (ou de plusieurs moyen(s) de l'atteindre. Il ne s'agit pas pour nous de faire de la démonstration de théorème, mais d'atteindre les objectifs fixés par nos utilisateurs.

Il peut exister des buts (que l'on qualifie généralement de buts internes), seulement visibles par le système.

Proposition **lien but-inférence** : Un but connaît l'ensemble des inférences qui permettent de le mener à bien.

Exemple : *Pour calculer la conversion appliquée à la charge (but : trouver la valeur de la conversion), on peut partir des conditions opératoires si on les connaît (à travers une inférence particulière), ou des qualités désirées des produits (à travers d'autres inférences)* .

Formellement, l'examen des relations permet de trouver celles qui permettent d'accomplir un but donné.

V.2.1.3) UTILISATION DYNAMIQUE DES CONNAISSANCES

Nous introduisons maintenant les notions de contexte, de problème, de tâche, pour représenter dynamiquement les connaissances. De même, l'exemple du paragraphe V.2.4.1 résume toutes les définitions introduites.

Définition du **contexte** : le contexte est un ensemble de valeurs.

Ces valeurs correspondent le plus souvent à des attributs d'objets, instances des classes du domaine.

Exemple : *un appel d'offre particulier, une fois saisi en machine, va devenir le contexte initial de notre étude. L'appel d'offre représente un ensemble de valeurs, qui alimenteront des flux dans le schéma d'inférence, ce qui permettra les raisonnements.*

Formellement, le domaine $V = \{V_1, V_2, \ldots, V_n\}$ est un espace de variables pouvant prendre des valeurs entières, réelles, booléennes, symboliques, etc., ayant pour domaines de définition $d_1, d_2, \ldots, d_n$, et le contexte est un point d'un sous-espace du domaine.

Définition d'un **problème** : Un problème (un résultat à obtenir) correspond à un but en opération, et à un contexte. Quand un utilisateur cherche à démontrer un but dans un contexte donné, il se pose un problème.

Exemple : *dans le cadre d'un appel d'offre particulier, on se pose le problème de déterminer la consommation d'hydrogène.*

Formellement, un problème P est la donnée d'un doublet (c, B),
où c est un contexte et B un but.

On nomme maintenant **IB**, l'ensemble de toutes les inférences qui correspondent à un but, c'est-à-dire dont l'un des flux de sortie est un but.
Définition d'une **tâche** : une tâche se définit comme le rapprochement d'une inférence, prise dans IB et d'un contexte.

Cette définition est conforme à nos remarques préliminaires (voir paragraphe V.2.1.1), qui réserve la notion de tâche à l'utilisation dynamique des connaissances. L'avantage de cette définition est de bien poser la distinction entre but et tâche. L'inconvénient est de réduire l'importance formelle de ce concept de tâche.

Puisqu'une tâche renvoie à une inférence, on définit également des tâches élémentaires et des tâches composées.

Formellement, une tâche T est la donnée d'un doublet (c, I),
où c est un contexte, et I une inférence prise dans IB.

Proposition **lien tâche problème** : une tâche permet de résoudre un problème.

Ce problème est déterminé comme suit : on considère l'inférence de la tâche, puis on cherche le flux de sortie de l'inférence, puis le but du flux (qui existe puisque l'inférence appartient à IB). Une fois qu'on a le but, on recherche le problème correspondant, parmi l'ensemble des problèmes.

Explication sur la propriété sortie d'une inférence : si une inférence avait plusieurs sorties possibles (soit N ce nombre), chacune d'entre elles pourraient détenir un but différent. Comme chaque problème correspond à un but, une même tâche pourrait participer à la résolution de N problèmes, ce que nous souhaitons éviter.

Pour résoudre un problème, il faut enchaîner un certain nombre de tâches, chacune faisant référence à un but donné. Chaque mise en oeuvre de tâche conduit à faire un pas dans le raisonnement, ce qui se traduit par l'activation d'une inférence. Et, chaque inférence, pour s'exécuter doit mettre en oeuvre une méthode.

De sorte, que si l'on observe le système en cours de résolution, c'est-à-dire dans sa dynamique, on parle de tâches et de problèmes, tandis qu'au niveau conceptuel, quand on analyse le système, statiquement, on parle de buts et d'inférences et de méthodes qui permettent de les atteindre. Cette façon de voir est conforme à l'idée que les ingénieurs se font de leur propre activité, qui parlent plutôt de problèmes et de tâches.

V.2.1.4) IMPLANTATION DES DÉFINITIONS

Une fois toutes ces définitions posées, il nous reste à les représenter sous forme d'objets. Nous pourrons ensuite réécrire ces derniers à l'aide d'un langage (à objets) particulier.

Dans les sections qui vont suivre V.2.2 et V.2.3, nous proposons une représentation 'objet', que nous détaillons méthodiquement. Notre description se veut indépendante des langages de programmation et nous laissons le soin au lecteur d'implanter ces modèles dans le langage de son choix.

Cependant, pour lever toute ambiguïté sur les notions que nous utilisons et notamment sur la notion de méthode, il nous faut préciser le langage de programmation qui finalement permet de faire tourner notre modèle : l'environnement que nous avons utilisé est celui de Smeci (marque déposée par Ilog S.A.), et le langage de programmation utilisé est Lelisp (développé par l'INRIA).

V.2.2) REPRÉSENTATION STATIQUE DES CONNAISSANCES DANS PRINCE

Le raisonnement s'appuie sur les deux notions de **relation** et de **flux** (correspondant essentiellement aux concepts d'inférence et de but). Nous détaillerons dans cette section la manière précise dont nous avons implanté ces concepts dans PRINCE, et **comment nous les avons modélisés sous forme d'objets**. L'idée de départ est d'indiquer au niveau de chaque pas de raisonnement quelles sont les données d'entrée et de sortie, pour que l'utilisateur puisse utiliser l'inférence ainsi décrite indépendamment des autres, ou l'enchaîner.

Distinguer but et inférence du point de vue opérationnel est justifié si :

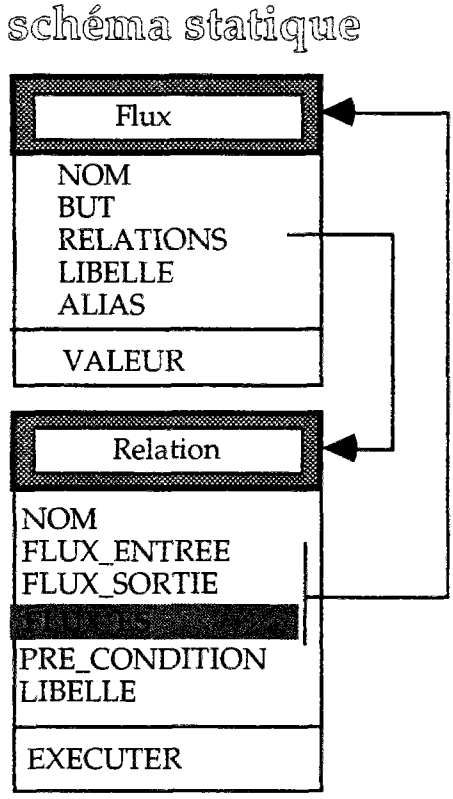
1) on souhaite donner la possibilité aux utilisateurs de manipuler les inférences seules (cas de PRINCE).
2) pour accomplir un but, l'ingénieur a à sa disposition plusieurs façons de faire différentes, dépendantes du contexte et doit décider, à chaque pas, quelle est la bonne méthode adaptée à chaque problème, chaque méthode faisant l'objet d'une inférence.

3) les enchaînements, pour parvenir aux buts qui ont été assignés par l'utilisateur, sont variables, dépendent des méthodes employées ou du contexte (cas de PRINCE).

V.2.2.1) LES FLUX ET LES RELATIONS

Les relations et les flux s'articulent dans PRINCE suivant le schéma n°4.

schéma 4 : **la représentation du raisonnement dans PRINCE, schéma statique**



Remarques :
• Nous avons signalé qu'un but est un flux de sortie d'une relation, (ce qui ne signifie pas qu'un flux de sortie est toujours un but).
• Une relation permet de définir les inférences.

Détail des flux (voir paragraphe V.2.2.3)

• NOM : (champ de type symbole)
Nom du flux, utilisé de façon interne

• LIBELLE : (champ de type chaîne de caractères)
Libellé plus explicatif

• BUT : (champ de type chaîne de caractères)
Chaque fois que l'obtention d'un résultat peut faire l'objet d'une demande de la part de l'utilisateur, ce champ est alimenté. D'autres buts internes de résolution existent, bien entendu, qui ne sont pas visibles par l'utilisateur. Dans ce cas, le champ BUT reste vide. Enfin, une vérification s'impose : ce champ ne peut correspondre qu'à des flux de sortie ou d'entrée-sortie.

• ALIAS : (champ de type symbole)
Nom mnémonique, utilisé dans les échanges avec l'utilisateur
exemple : *TSEJ* pour Temps de SÉJour.

• RELATIONS : (champ de type liste d'objets de la classe relation)
Ce champ contient la liste des relations qui permettent de déterminer ce flux. L'examen des relations permet d'alimenter ce champ automatiquement : on parcourt l'ensemble des relations et on extrait toutes celles qui ont le flux étudié en entrée, ou en entrée-sortie.

• VALEUR : (champ de type méthode)
Quand l'utilisateur lance un raisonnement qui enchaîne une suite de buts, il faut donner une valeur aux flux d'entrée et reporter les valeurs trouvées, s'il y a correspondance, dans le domaine d'application. C'est la fonction d'une méthode (au sens des langages à objets) VALEUR, qui s'applique aux flux.

Détail des relations (voir paragraphe V.2.2.4):

• NOM, LIBELLÉ : même chose que pour les flux.

• FLUX_ENTRÉE, FLUX_SORTIE, FLUX_ES: (champs de type liste d'objets de la classe flux)
Ce champ désigne l'ensemble des objets flux en entrée, sortie ou entrée-sortie de la relation.

Quand on signifie qu'une relation dans l'application PRINCE est inversible par rapport à un paramètre, cela veut dire qu'en appliquant la méthode d'exécution inverse, on peut par exemple, pour une relation qui a 3 paramètres d'entrée E1, E2 et E3 et une valeur de sortie S1, retrouver la valeur de E1, à partir de la connaissance de E2, E3 et S1, si cette relation est inversible par rapport à E1. C'est bien souvent le cas des relations courbes et formules. Ceci représente bien une notion (mathématique) d'inversion, et non d'ailleurs un concept d'abduction. La représentation adoptée permet, à l'aide de la description d'une seule relation, de mentionner cette possibilité d'inversion. C'est la raison de la présence des variables de type entrée-sortie (FLUX_ES, ES pour entrée-sortie) au niveau des relations.

• PRE_CONDITION
Ce champ permet de préciser l'ensemble des conditions d'exécution et de lancement des inférences sous-jacentes. Il s'agit d'un prédicat, qui porte sur les variables présentes dans le contexte, quand ce dernier est défini (pour la

description du contexte, se reporter au schéma dynamique, section V.2.3.1). Quand le contexte est indéfini, comme dans le cas d'utilisation isolée des relations, les conditions ne peuvent porter que sur les valeurs des flux d'entrée et de sortie.

• EXECUTER : (champ de type méthode)
Méthode d'exécution de la relation

Quand l'utilisateur veut utiliser une inférence de façon isolée, il fournit les valeurs d'entrée nécessaires, et lance la méthode EXECUTER. Cette dernière déduit quelle est l'inconnue, applique un calcul direct ou inverse suivant le cas, et fournit le résultat (ce qu'il n'aurait pas été possible de faire aussi synthétiquement avec une classe Inférence).

Les types de relations entre les flux d'entrée et de sortie sont assez peu nombreux : on trouve pour l'essentiel des formules, des règles, des programmes et des corrélations graphiques. Les méthodes EXECUTER sont construites de façon générique par type (ou classe de relation), comme nous le verrons au paragraphe V.2.2.4.

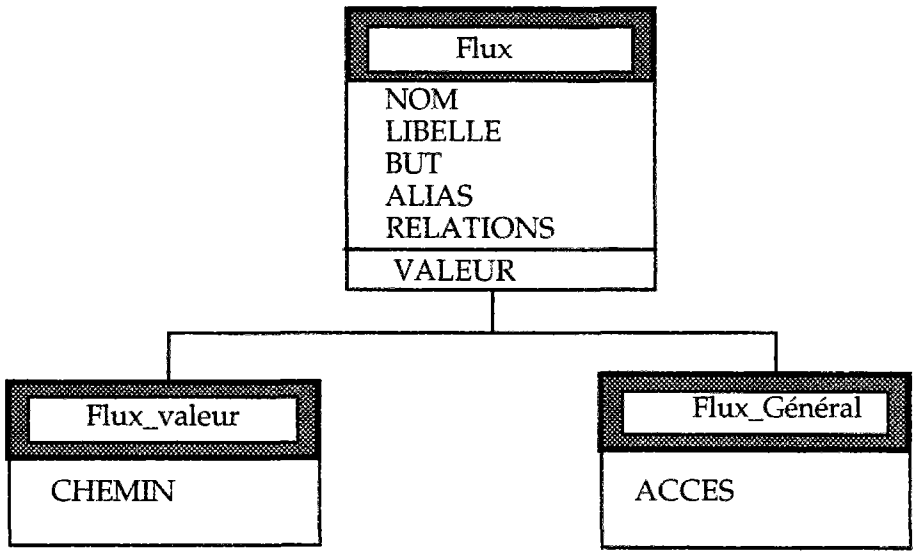
Dès qu'une relation est inversible, cette faculté d'inversion est donnée aux utilisateurs. La description détaillée des différents algorithmes d'inversion est exposée dans le chapitre VI *courbes*, car la visualisation des courbes inverses est une des fonctionnalités très appréciées des utilisateurs.

V.2.2.2) DIFFÉRENTS TYPES DE FLUX

Les relations connaissent leurs flux d'entrée et de sortie. Ces flux, quand ils sont utilisés en tant qu'entrée sont munis d'une méthode (VALEUR) qui permet d'aller chercher leurs valeurs (quand elles sont disponibles) à travers le schéma objet du domaine. A l'inverse, quand il s'agit de résultats, la même méthode VALEUR reporte les valeurs trouvées dans le domaine, si il y a correspondance.

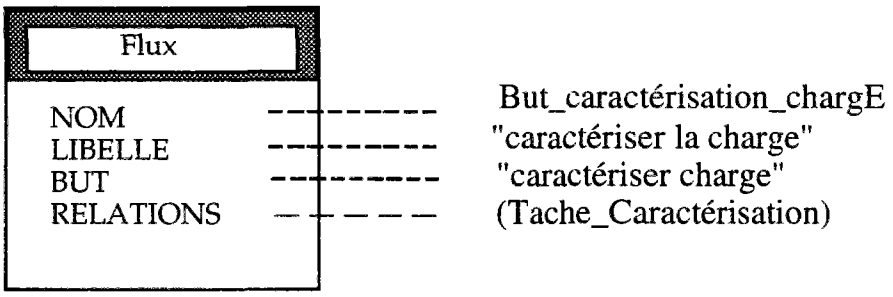
Les différentes façons d'être en correspondance avec le domaine conduisent à définir une hiérarchie de plusieurs classes de flux (au sens des langages à objets).

schéma 5 : **hiérarchie des classes de flux**



• Si nous avons affaire à des buts simples, du genre *'caractériser la charge'*, dont le flux correspondant n'a pas de valeur de sortie dans le domaine, la classe correspondante sera celle des flux simples :

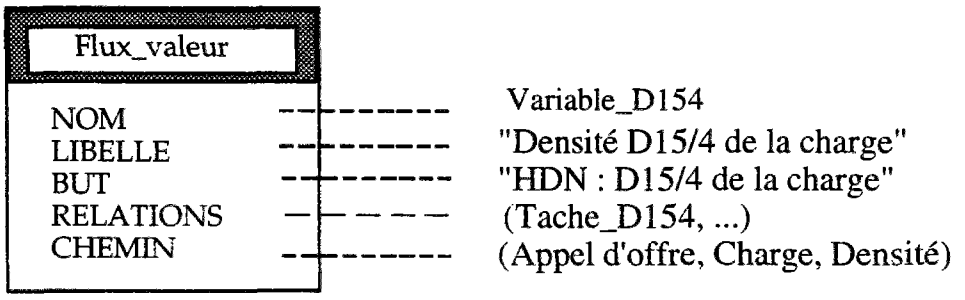
schéma 6 : **classe flux**



• avec des buts du style *'trouver la valeur du soufre de l'effluent'* qui, eux, sont en correspondance avec le domaine, la description objet sous-jacente va contenir une référence à la classe et au champ de l'objet qu'ils sont censés désigner, par l'intermédiaire d'un chemin qui permettra d'aller rechercher l'objet adéquat dans le contexte (ou de reporter sa valeur).

Exemple de chemin : *Pour rechercher la densité de la charge, le chemin proposé est (Appel_Offre, CHARGE, DENSITE), c'est-à-dire qu'on cherchera d'abord un objet de classe Appel_Offre dans le contexte, puis l'ayant trouvé, on en prendra le champ CHARGE, qui référence l'objet charge du cas traité, dont on extraie enfin le champ DENSITE. Fort heureusement, il n'y a jamais qu'un appel d'offre par jeu de simulation, ce qui lève toute ambiguïté dans le chemin.*

schéma 7 : **classe Flux_valeur**



• Il y a ainsi autant de classes de flux que de moyens de piocher une ou des valeurs dans le contexte. A la limite, il est très simple de construire la classe la plus générale, qui aurait pour attribut supplémentaire, par rapport aux flux simples, le champ ACCES, qui désigne le nom de la fonction d'accès qu'il faut déclencher pour explorer le contexte ou reporter une valeur dans le domaine.
• Rappel : si un flux n'est pas un but, son champ BUT n'est pas alimenté.

V.2.2.3) DIFFÉRENTS TYPES DE RELATIONS

Classiquement, nous disposons de deux types de relations de base : les relations terminales qui ne peuvent être qu'exécutées, et les relations de contrôle, correspondant au connecteur classique 'et'. Au cours de l'exécution, ces relations dans un contexte donné permettront de définir des tâches, et la typologie des tâches et des relations sera la même.

V.2.2.3.1) Les relations de contrôle

Dans certains cas, un but peut être a priori décomposé en sous-buts. Cet enchaînement correspond à une structure de contrôle séquentiel, modélisé par le connecteur 'et'.

Quand nous avons à choisir entre plusieurs méthodes possibles en fonction du contexte d'exécution, ce sont les pré-conditions qui éliminent les méthodes inadéquates. Le connecteur 'ou' n'est donc pas nécessaire.

Il faut remarquer que la description du moteur, telle que nous l'avons résumée à la section V.2.3.1, ne fait jamais référence à la façon d'exécuter telle ou telle tâche. C'est la tâche qui sait elle-même comment elle doit être lancée. Ainsi, pour le moteur, exécuter une tâche composée, ou une tâche élémentaire ne donnera lieu à aucune différence de comportement. C'est la méthode associée à chaque tâche qui sera différente.

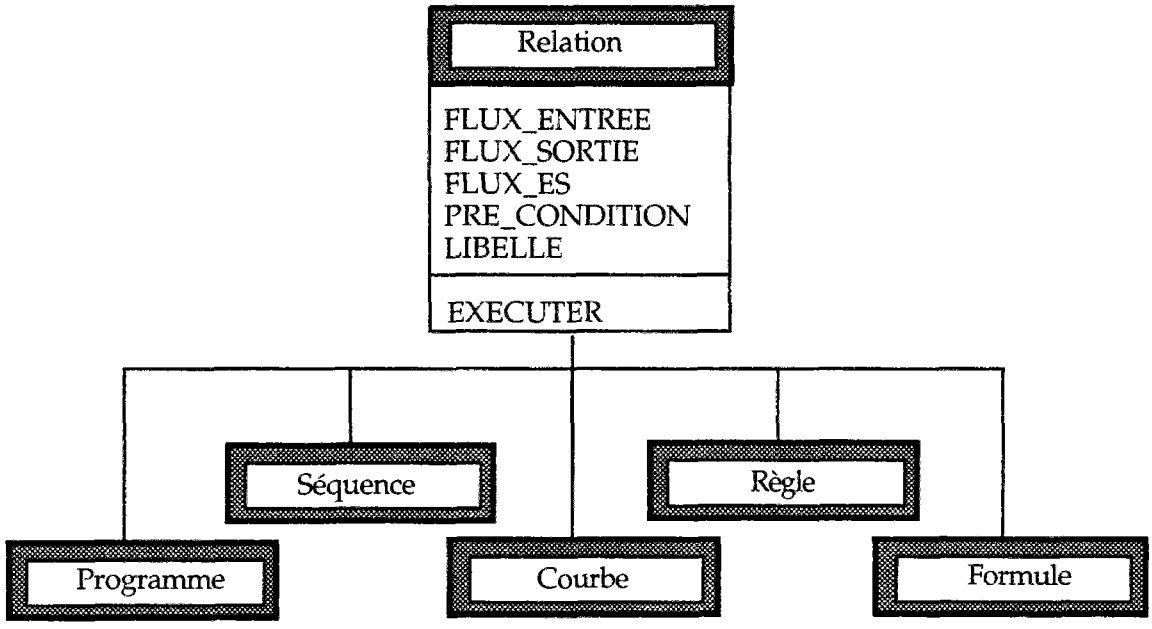
V.2.2.3.2) Méthodes sous-jacentes aux relations

Nous avons dégagé quelques types de méthodes correspondant chacune à un mode d'exécution. Ces différents types de méthode donnent lieu à différentes classes de relations.

- Les relations '*séquence* ' activent séquentiellement une suite d'inférences.
- Les relations '*règle* ' se définissent par l'intermédiaire de règles expertes et leur résolution s'appuie sur le fonctionnement d'un moteur d'inférences.
- Les relations '*programme* ' lancent des programmes ou des procédures;
- Les relations '*formule* ' sont basées sur la donnée d'une certaine formule
- Les relations '*courbe* ' exploitent un réseau d'abaques, dont chaque courbe est donnée par une liste de points.

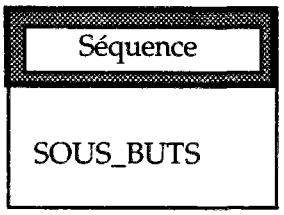
Les relations que nous venons d'évoquer sont basées sur la hiérarchie suivante, qui découle d'une classe racine Relation :

schéma 8 : **hiérarchie des classes de relation**



La méthode EXECUTER est redéfinie pour chaque type de relation : (Séquence, Programme, Courbe, Formule, Règle). Préalablement au déclenchement de cette méthode, on constitue la liste de doublets, comprenant l'alias des flux d'entrée et leur valeur, recherchée par l'intermédiaire de la méthode VALEUR. Cette liste constitue l'ensemble des paramètres d'appel de la méthode. Rappelons que les méthodes inverses sont décrites au chapitre VI *courbes.*
Exemple : *((TSEJ, valeur de TSEJ) (PRESSION, valeur de PRESSION) ...).*

• **Séquence**
schéma 9 : **classe séquence**

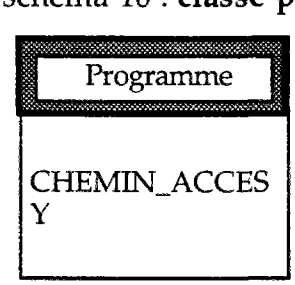


• SOUS_BUTS : (champ de type liste d'objets de la classe flux)
Ce champ contient la liste des sous-buts qui doivent être atteints successivement, puisque la relation *Séquence* permet la décomposition d'un but en sous-buts et leur enchaînement en séquence, au moment de l'exécution.

La méthode EXECUTER associée lit les buts référencés par le champ SOUS_BUTS, et pour chacun d'eux crée le problème correspondant à chacun d'eux, dans l'ordre où ils sont indiqués (voir la partie dynamique, section V.2.3, pour le détail des objets de la classe problème).

A titre d'illustration, le but 'CARACTÉRISER LA CHARGE' se décompose en deux sous-buts :
- vérifier que toutes les valeurs des propriétés de la charge sont dans des bornes admissibles (but 'VERIFIER_CHARGE', la valeur de l'azote ne doit pas être trop élevée, par exemple), et
- s'assurer de la cohérence de la charge (but 'CAS_NORMAL'), c'est-à-dire vérifier que les valeurs des propriétés sont cohérentes entre elles.

• **Programme**
schéma 10 : **classe programme**



La relation Programme contient, outre les champs classiques de la catégorie mère, des champs spécifiques.

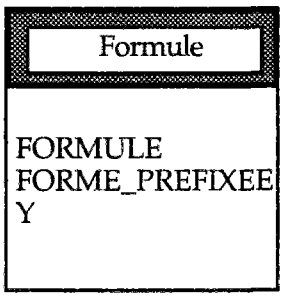
• CHEMIN_ACCES : (champ de type liste de chaîne de caractères)
Ce champ permet de charger le programme sous-jacent à la relation. Pour nous, il se décompose en trois de la façon suivante : (répertoire, fichier source contenant le programme, nom du programme). La description du chemin d'accès sous forme de liste donne la possibilité de s'adapter à n'importe quel système d'exploitation.

• Y : (champ de type symbole)

Le champ Y permet de savoir quelle est la sortie normale du programme.

La liste des valeurs d'entrée est reçue comme paramètre du programme, qui commence bien sûr par l'analyser avant de s'exécuter.

• **Formule**
schéma 11 : **classe formule**

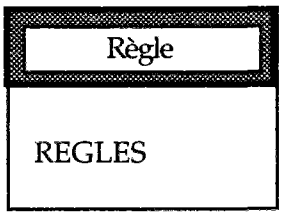


• FORMULE : (champ de type chaîne de caractères)
Le champ FORMULE de la classe Formule contient la formule telle qu'elle est saisie (par exemple $HDN = (N_C - N_e) / N_C$).

• FORME_PREFIXEE : (champ de type liste de symboles)
La formule, dès qu'elle est saisie, est mise immédiatement sous forme préfixée (voir annexe du chapitre VI *courbes*) pour gagner du temps à l'exécution.

La méthode EXECUTER attribue une valeur aux différents littéraux qui interviennent dans la formule, puis l'évalue.

• **Règle**
schéma 12 : **classe règle**

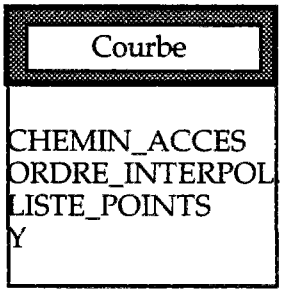


La relation Règle mentionne uniquement les règles (contenues dans le champ REGLES) qui doivent être déclenchées pour pouvoir s'exécuter. En fait, pour le moment dans PRINCE, aucune valeur n'est fournie en sortie par ce type de tâches.

Les règles, contrairement aux autres tâches dans PRINCE, ne s'utilisent pas de façon isolée. Si certaines règles fournissaient des valeurs, ce mode d'utilisation pourrait cependant s'envisager. Enfin, l'inversion de telles tâches est délicat : une règle s'emploie plutôt quand le domaine de définition des valeurs d'entrée est

discret, ou quand les définitions des fonctions qui donnent les valeurs de sortie, présentent des discontinuités

• **Courbe**
schéma 13 : **classe courbe**



• CHEMIN_ACCES
Même chose que pour les programmes. Ces champs permettent de référencer le fichier des points de définition.

• LISTE_POINTS : (champ de liste de valeurs)
Ce champ contient la liste des points de définition, une fois chargés en mémoire. Ces points ne sont chargés que si nécessaire, pour ne pas encombrer.

• Y même chose que pour les programmes.

• ORDRE_INTERPOLATION : (champ de type liste de symboles)
Cet attribut permet de préciser l'ordre dans lequel les différents paramètres d'entrée de la méthode exécuter doivent être interpolés. C'est en effet nécessaire pour deux raisons : d'abord, une courbe n'est pas nécessairement inversible par rapport à tous les paramètres, et ensuite, les interpolations conduisent de façon intrinsèque à des erreurs numériques et ces erreurs peuvent être parfois minimisées en suivant l'ordre indiqué : en effet, interpoler est toujours moins précis que suivre une courbe exacte. Il vaut mieux interpoler d'abord les paramètres dont les variations sont représentées par le plus grand nombre de points et il vaut mieux interpoler d'abord les paramètres dont les échelles de variation sont les plus étroites.

Dans son principe, la méthode EXECUTER interpole successivement les points de base lus dans le champ LISTE_POINTS dans l'ordre donné par l'attribut ORDRE_INTERPOLATION.

V.2.2.4) Deux conséquences de notre représentation du raisonnement

Notre représentation du raisonnement dans PRINCE permet aux utilisateurs de l'application, experts ou cogniticiens, de bénéficier de deux facilités importantes.

- principe de modularité

Il est très facile dans cette optique, de **rajouter tous les nouveaux types de relations** nécessaires, en liaison avec l'application que l'on est en train de construire : une fois la structure de la nouvelle relation définie, il suffit de lui fournir une méthode EXECUTER adaptée (et une méthode inverse si nécessaire). On aurait pu, par exemple, bâtir ainsi une relation 'résolution d'un système d'équations linéaires'.

En ce sens, PRINCE est un générateur de systèmes à base de connaissances, quand celles-ci s'expriment sous forme de tâches. Ceci dit, les méthodes EXECUTER peuvent être complexes et sophistiquées, et c'est bien évidemment au responsable du système informatique de les écrire, et de les tester. Du point de vue de la validation, une fois qu'une telle méthode fonctionne, elle ne peut qu'être difficilement remise en cause, et ce sont les connaissances portées par les tâches qui seront mises en doute, plutôt que le fonctionnement de l'application. Le partage des responsabilités, entre experts et cogniticiens, est facile.

- principe de maintenance

C'est là la seconde conséquence. L'application est rendue facilement maintenable. Par exemple, il est très facile pour un expert autorisé d'intervenir sur le fichier de points d'une relation 'courbes', ou de remettre en cause une formule, sans qu'on ait besoin de modifier en quoi que ce soit l'application elle-même.

Pratiquement, les quelques relations de base que nous avons décrites, sont le support de l'essentiel de la connaissance dans PRINCE. Dans cette base de connaissances, les relations à base de courbes, représentant des liaisons graphiques entre des variables, sont, par exemple, très souvent mises à contribution.

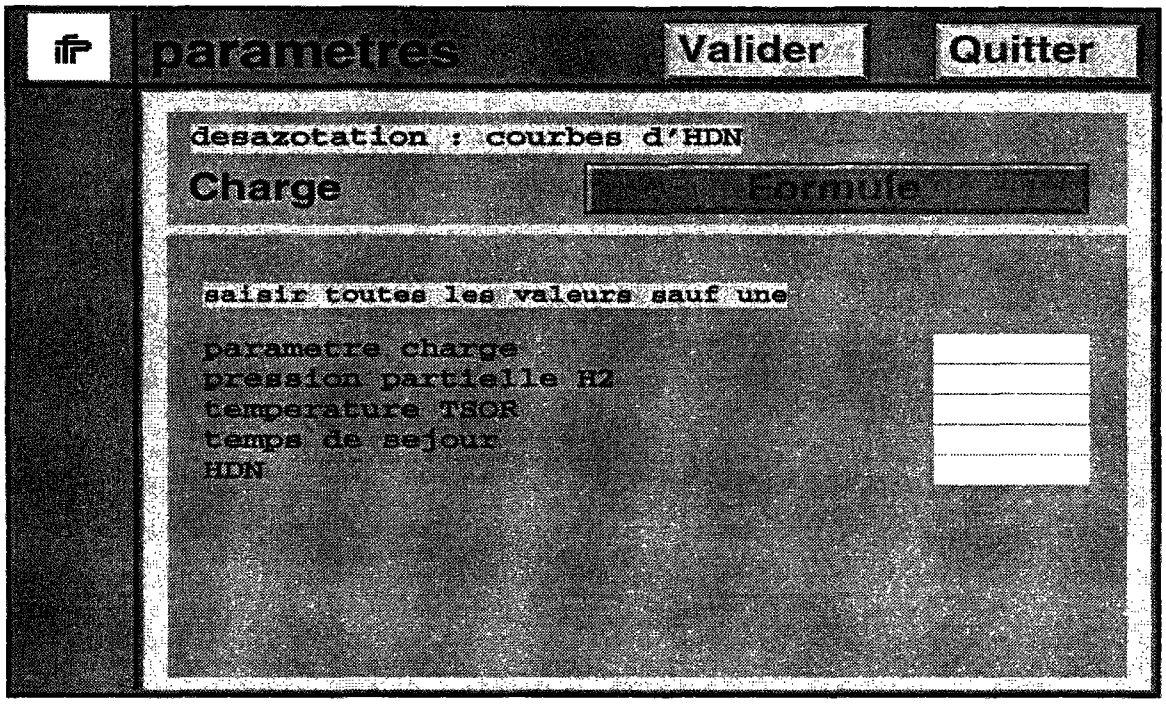
Rien ne s'oppose, dans cette optique, à ce que ce l'expert lui-même crée les tâches qui lui sont nécessaires, s'il ne s'agit pas d'un nouveau type de tâches. Encore que cela ne soit pas encore fait dans PRINCE.

Pour valider ses modifications, l'expert a besoin d'outils spécifiques. Dans PRINCE, cette fonction est réalisée pour l'essentiel par des visualisations, qui se calculent par l'intermédiaire de la méthode CALCULER_COURBES, qui sera développée dans le chapitre VI *courbes*. La structuration en tâches que nous proposons s'adapte particulièrement bien à ce traitement.

V.2.2.5) UTILISATION DE FAÇON ISOLÉE

Si l'utilisateur se pose une question isolée, il dispose déjà naturellement de toutes les relations du schéma conceptuel du raisonnement dans le domaine. Chaque relation connaissant ses variables d'entrée et de sortie, l'application propose à l'utilisateur un formalisme systématique qui permet à ce dernier de fournir les valeurs d'entrée nécessaires.

schéma 14 : **utilisation des tâches de façon isolée, exemple de l'HDN**

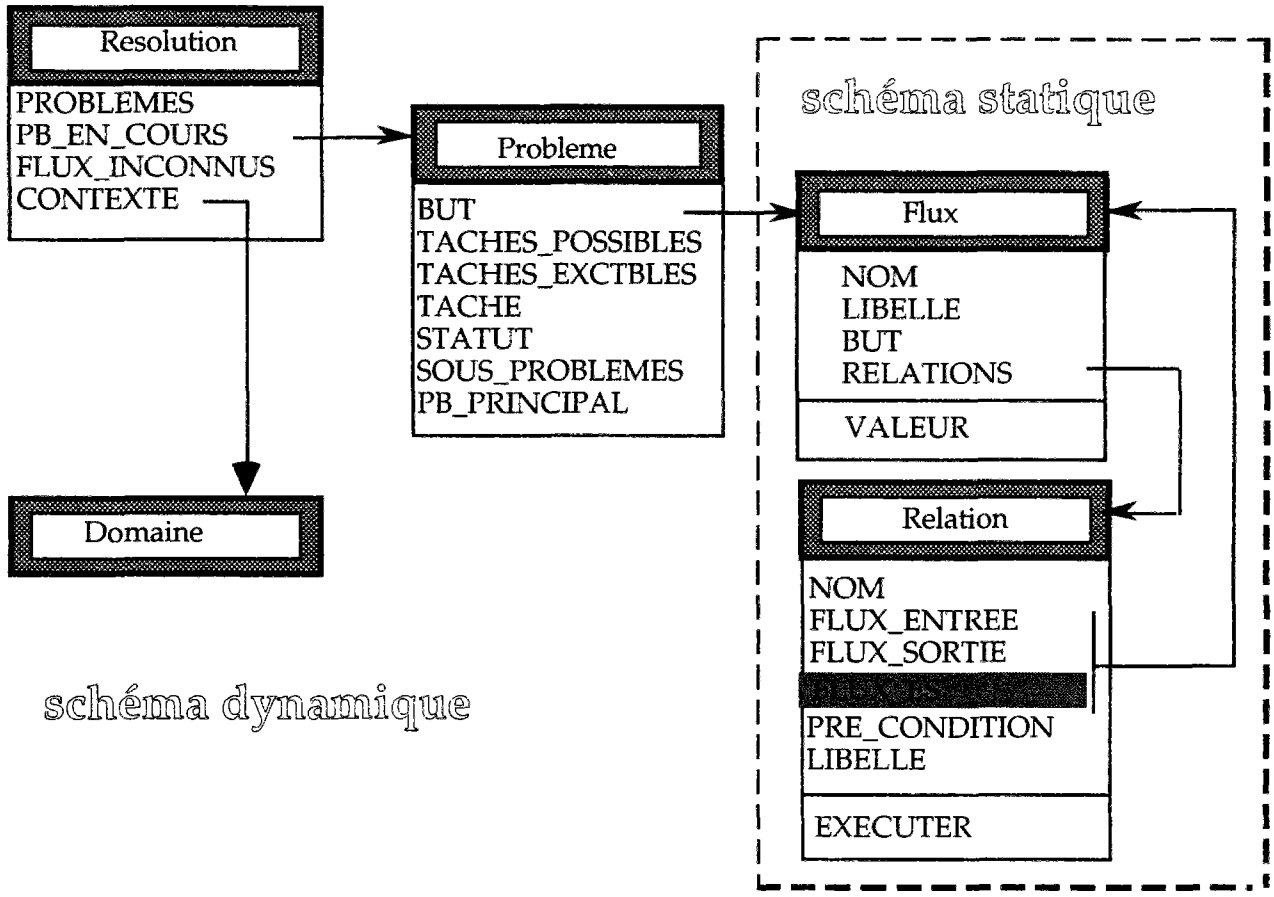


Exemple d'écran de lancement de tâche basée sur une relation inversible. La zone formule a un usage spécifique.

Par exemple, *sur cet écran , on fournit les valeurs du paramètre de charge, de pression partielle H2, de température et de temps de séjour et le système calcule alors le résultat, c'est-à-dire l'HDN. Inversement, on aurait pu fournir l'HDN, et laisser libre le temps de séjour, et le système l'aurait déterminé.*

V.2.3) LA RÉSOLUTION DES PROBLÈMES DANS PRINCE

schéma 15 : **résolution des problèmes dans PRINCE, schéma dynamique**



La résolution des problèmes soumis à l'application PRINCE par les utilisateurs sera suivie par l'intermédiaire d'un objet appelé RésolutioN (les majuscules en tête et en fin de mot sont dans ce cas volontaires : il faut savoir distinguer les ObjetS comme l'objet RésolutioN, la Classe comme la classe Résolution, l'ATTRIBUT comme l'attribut CONTEXTE et une simple mention dans une phrase).

V.2.3.1) PROBLÈMES

L'utilisateur choisit une liste de buts. Chaque but conduit à la création d'un objet problème, et chaque objet problème ainsi créé va s'ajouter dans le champ PROBLÈMES de l'objet RésolutioN.

Les objets problèmes reflètent les problèmes réels que se posent les utilisateurs et vont évoluer naturellement durant la résolution. Au début, poser un problème, c'est se donner un but, la détermination d'une valeur de sortie par exemple. Ensuite, la première étape est de rechercher quelles sont les inférences susceptibles d'établir ce but. Après avoir écarté les inférences manifestement

inadéquates dans le contexte du problème, c'est-à-dire celles pour lesquelles il manque au moins une valeur d'entrée, ou celles qui ne peuvent s'appliquer dans ce contexte à cause de leurs conditions d'exécution, l'une d'entre elles est choisie, et exécutée. Si l'exécution réussit, on obtient ainsi la solution du problème, qui est renvoyée au domaine d'application si nécessaire.

Un problème sera résolu, quand son but sera atteint. Deux cas peuvent se présenter, suivant la nature des buts en question, sans que le comportement de l'application soit différent d'ailleurs : Si un problème concerne la recherche de valeurs, significatives au niveau du procédé, comme '*trouver la valeur de la densité de l'effluent*' ', le but est atteint et le problème résolu, quand le résultat est trouvé. Sinon, aucune valeur de sortie n'est présente ni produite au niveau de la tâche (si ce n'est que le but est accompli) : par exemple, le but '*caractérisation de la charge*' conduit à émettre une suite d'appréciations sur la charge (le pétrole), qui se formalisent finalement par un texte de diagnostic à l'écran.

Détail de la classe Problème

- BUT : (champ de type objet de la classe flux)
  Ce champ renvoie à l'objet but sous-jacent au problème. Rappelons que les objets but sont également de la classe flux.

- TACHES_POSSIBLES : (champ de type liste d'objets de la classe tâche)
  Les tâches possibles sont lues sur l'objet but trouvé. En fait, comme nous sommes au cours de la résolution, les inférences susceptibles d'atteindre le but demandé, rapportées au contexte particulier de la résolution, deviennent des tâches (voir les définitions). Les tâches possibles du problème sont déduites des inférences détenues par le but.

- TACHES_EXECUTABLES : (champ de type liste d'objets de la classe tâche)
  Si les conditions d'exécution d'une tâche possible sont remplies, elle devient une tâche exécutable, et sa référence est ajoutée automatiquement aux autres références de ce champ.

- TACHE : (champ de type objet de la classe tâche)
  C'est la tâche finalement retenue pour résoudre le problème. C'est au départ l'une des tâches des TACHES_EXECUTABLES. La façon de la choisir est expliquée en section V.2.3.2.

- STATUT : (champ de type symbole)
  À la création du problème, ce champ est vide. Quand une tâche a été sélectionnée (champ TÂCHE $\neq$ vide), et que cette dernière est lancée, le STATUT passe à 'en-cours'. Quand l'exécution a réussi, ce champ passe à 'résolu', comme le problème sous-jacent.

- SOUS_PROBLÈMES : (champ de type liste d'objets de la classe problème).
- PB_PRINCIPAL : (champ de type objet de la classe problème)
  Ces deux champs servent à suivre les problèmes correspondant à des tâches de type séquence, et qui se décomposent finalement en une suite de sous-

problèmes. Ce champ se distingue du champ PB_EN_COURS de l'objet RésolutioN : un problème principal PB peut avoir plusieurs SOUS_PROBLÈMES PB_1 et PB_2. Quand on est en train de traiter PB_2, le champ PB_EN_COURS vaut PB_2. Le champ PB_PRINCIPAL de PB_2 vaut PB.

V.2.3.2) RÉSOLUTION

Détail de la classe Résolution

- CONTEXTE : (champ de type liste d'objets dépendant du domaine)
  Cette résolution s'appuie sur un contexte, qui contient l'ensemble des valeurs du cas d'étude traité. Plus exactement, le contexte fait référence à tous les objets propres au cas d'étude et désigne la partie dynamique de l'application, puisque ce contexte change avec chaque étude, et évolue au cours de la résolution. Chaque fois que le moteur aura besoin d'une valeur particulière, il ira piocher dans le contexte. Inversement, les nouvelles valeurs déduites par le moteur seront reportées dans le contexte.

- PROBLÈMES : (champ de type liste d'objets de la classe problème)
  Chaque problème posé par un utilisateur se concrétise par la création d'un objet problème. La liste des problèmes est stockée dans le champ PROBLÈMES de l'objet RésolutioN.

- PB_EN_COURS : (champ de type objet de la classe problème)
  Au cours de la résolution, le premier problème de la liste des problèmes devient le problème en cours.

- FLUX_INCONNUS : (champ de type liste d'objets de la classe flux)
  Chaque fois qu'un but ne peut pas être atteint, il est ajouté à ce champ, ce qui évite de se reposer la même question tout le temps.

DESCRIPTION DE L'ALGORITHME

- 1) Appelons ProblèmE le problème en cours. On lit le champ BUT de ProblèmE, qui renvoie sur l'objet but correspondant; appelons-le B. Ce but B contient la liste des relations (champ RELATIONS) qui ont ce but en sortie. Ces relations sont reportées au niveau de ProblèmE. Elles deviennent alors des tâches (champ TACHES_POSSIBLES de l'objet PB) : une tâche est en effet le rapprochement d'un but, d'une inférence et d'un contexte (voir les définitions), et un problème, par construction, réalise ce lien. En particulier, l'inférence se déduit de l'association du but et de la relation.

- 2) On commence par ne retenir dans TACHES_POSSIBLES que les tâches dont toutes les données (ou flux) d'entrée sont présentes.

• 3) On applique les conditions d'exécution aux tâches de TACHES_POSSIBLES. Celles dont les conditions d'exécution (lues sur le champ PRE_CONDITION) sont vérifiées, sont basculées dans le champ TACHES_EXECUTABLES.

• 4) Parmi les tâches exécutables, susceptibles de réaliser le but B, on en sélectionne une, qui s'inscrit dans le champ TACHE de PB. Ce choix sera détaillé en 4.1. Cette tâche est lancée. Après l'exécution de la tâche, le problème est déclaré résolu (valeur qui est donnée au champ STATUT de l'objet PB). On enlève alors PB de la liste des problèmes (champ PROBLÈMES) de l'objet RésolutioN, et le problème suivant devient le problème en cours. La résolution s'arrête quand RésolutioN ne référence plus de problèmes.

• 4.1) Pour choisir une tâche, si plusieurs sont disponibles, on peut imaginer quantité de façons de faire différentes. On peut prendre la première tâche dans la liste, ou demander l'avis de l'utilisateur, ou encore utiliser des heuristiques si elles sont disponibles, qui peuvent tenir compte du contexte, des difficultés des traitements à venir, des temps de calcul prévisibles, de la confiance des experts. À ce niveau, il est facile d'introduire une évaluation a priori des tâches

Dans PRINCE, cette situation n'arrive pas : plusieurs tâches sont souvent disponibles a priori pour atteindre un but, mais dans un contexte donné, il ne reste qu'une possible, quand on a examiné les flux d'entrée.

• 5) Si aucune tâche n'est finalement sélectionnée pour un problème donné, on débute un chaînage arrière. Plus précisément, chaque fois que la valeur d'un flux sera inconnue, on se posera le problème de déterminer cette valeur, et l'algorithme de résolution fonctionnera de manière identique pour ces nouveaux problèmes.

L'algorithme est résumé à la page suivante, avec les notations suivantes :
ObjeT (majuscule en début et en fin de mot) désigne un objet
CHAMP désigne le champ d'un objet
CHAMP(ObjeT) désigne la valeur prise par le champ CHAMP de l'objet ObjeT
+ (-) signifie ajouter à (retirer de) la liste
$A \cap B$ désigne l'intersection de A et B
$\in$ est le symbole d'appartenance

algorithme de résolution

```
TANT QUE PROBLÈMES(RésolutioN) ≠ vide
        soit ProblèmE = PB_EN_COURS(RésolutioN)
(1)     • FLUX_INCONNUS(RésolutioN) = FLUX_INCONNUS(RésolutioN)
                                              + BUT(ProblèmE)
        • TACHES_POSSIBLES(ProblèmE) = RELATIONS(BUT(ProblèmE))

(2)     POUR chaque tâche TâchE de TACHES_POSSIBLES(ProblèmE),
            soit LFE la liste des flux d'entrée de TâchE
            SI LFE ∩ FLUX_INCONNUS(RésolutioN) ≠ vide
                • éliminer cette tâche;
            SINON
                SI ∃FluX ∈ LFE, VALEUR(FluX) = vide
                    •TACHES_POSSIBLES(ProblèmE) =
                        TACHES_POSSIBLES(ProblèmE) - TâchE
                FIN SI
            FIN SI
        FIN POUR

(3)     POUR chaque tâche TâchE de TACHES_POSSIBLES(ProblèmE),
            TACHES_POSSIBLES(ProblèmE) =
                        TACHES_POSSIBLES(ProblèmE) - TâchE
            SI PRE_CONDITION(TâchE ) = vide ou
                (PRE_CONDITION(TâchE )≠ vide et
                 EXCUTER(PRE_CONDITION(TâchE )) ≠ faux),
                    • TACHES_EXECUTABLES(ProblèmE) =
                        TACHES_EXECUTABLES(ProblèmE) + TâchE
                FIN SI
            FIN SI
        FIN POUR

(4)     TANT QUE  STATUT(ProblèmE) ≠ résolu et
                  TACHES_EXECUTABLES(ProblèmE) ≠ vide,
(4.1)       • choisir une tâche TâchE ∈ TACHES_EXECUTABLES(ProblèmE)
            • soit RESULTAT = EXECUTER(TâchE)
            SI RESULTAT ≠ vide,,
                • STATUT(ProblèmE) = résolu
                • VALEUR(BUT(ProblèmE)) =  RESULTAT
                • FLUX_INCONNUS(RésolutioN) =
                        FLUX_INCONNUS(RésolutioN) - BUT(ProblèmE)
            FIN SI
        FIN TANT QUE

(5)     SI STATUT(ProblèmE) ≠ résolu
            • créer autant de nouveaux problèmes que de flux inconnus
            • FLUX_INCONNUS(RésolutioN) =
                        FLUX_INCONNUS(RésolutioN) + chacun de ces flux
        FIN SI

FIN TANT QUE
```

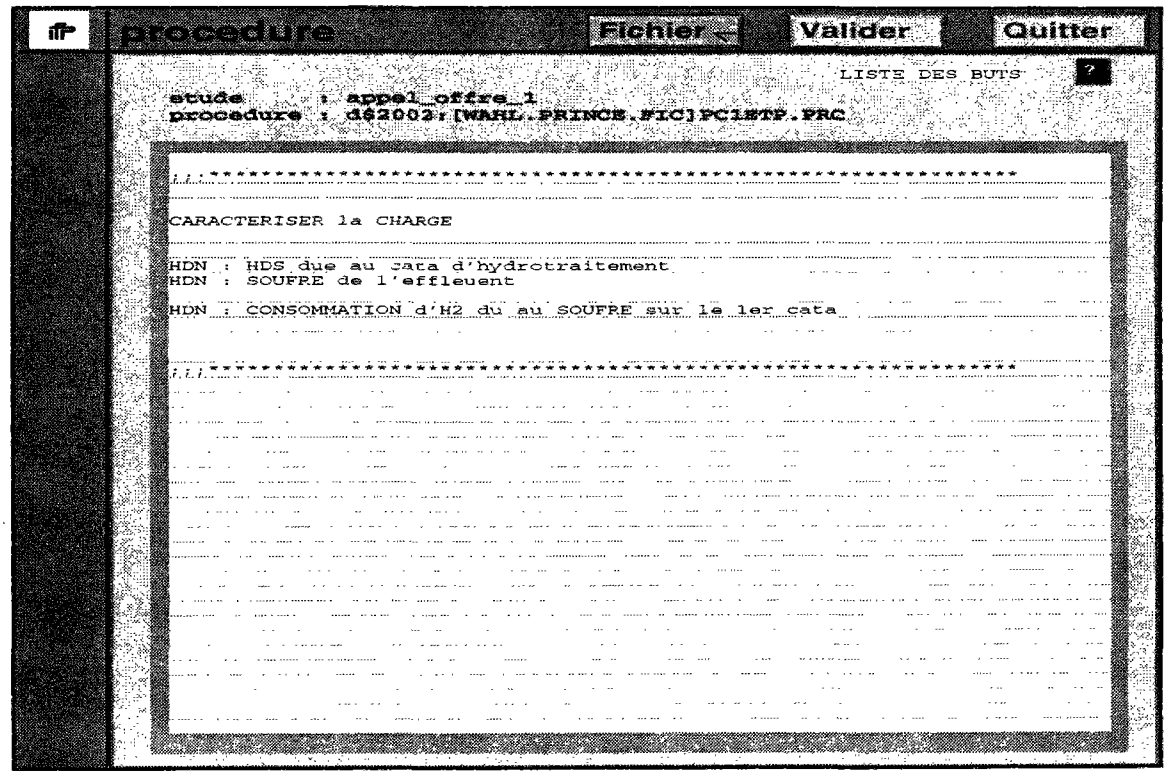
V.2.3.3) ENCHAÎNEMENT AUTOMATIQUE DES TÂCHES

Comme nous l'avons constaté, l'obtention des résultats suit dans un problème de spécification une démarche de type chaînage arrière. Par exemple pour calculer le soufre de l'effluent, il faut connaître le soufre de la charge et le taux de désulfuration. Si l'on n'a aucune idée de ce dernier, on examine l'ensemble des tâches qui permettent sa détermination. Les tâches sélectionnées ne seront déclenchées que si toutes leurs données sont présentes. Si pour fixer les idées on doit disposer de la température du réacteur, et que cette dernière est inconnue, on va essayer de la déterminer. Et l'on se retrouve confronté à nouveau à un raisonnement du type précédent (pour évaluer la température du réacteur, quelles sont les tâches possibles ...).

Ayant mis en évidence par l'analyse conceptuelle quelles sont les tâches et les variables, et quels sont les liens qui les unissent, cette démarche a pu être reproduite sans difficulté dans l'architecture proposée. Ainsi, ne pas décrire tous les enchaînements nécessaires pour résoudre l'ensemble de tous les problèmes de spécification possibles, est alors faisable, ce qui, on l'a vu, était une des complexités du domaine qui nous concerne. Le système sait reconstruire la démarche automatiquement.

Si l'utilisateur souhaite enchaîner plusieurs problèmes, il lui suffit de fournir au système une suite de buts (qu'on appelle procédure). Le langage de commande, adapté à cette fin, autorise la fourniture d'une suite de commandes sous forme textuelle. Le plus facile d'ailleurs est de modifier une procédure existante.

schéma 16 : **utilisation des tâches de façon enchaînée**



L'exemple qui est donné ici permet de *calculer la consommation d'hydrogène due au soufre*. Chaque ligne de cette procédure est un but, choisi parmi la liste des buts possibles (bouton point d'interrogation), sauf les lignes précédées d'un point virgule, qui sont considérées comme des commentaires, laissés libres pour l'utilisateur. L'utilisateur peut se constituer ainsi les procédures qu'il souhaite, les commenter, et les stocker pour les réutiliser ultérieurement.

Il est malheureusement facile pour l'utilisateur de se tromper dans l'ordre des lignes de la procédure. Dans ce cas, le système tente de lui-même de corriger cette situation anormale, en lançant autant de problèmes qu'il est nécessaire pour suivre la procédure, jusqu'à ce qu'il ne puisse plus résoudre aucun problème.

Enfin, si l'utilisateur préfère que le système fonctionne de façon autonome, le système dispose de toutes les informations qui lui sont nécessaires, pour reconstruire un chemin de résolution par chaînage arrière. PRINCE s'arrêtera quand tous les problèmes auront été examinés.

V.2.4) EXEMPLE SIMPLE

V.2.4.1) DÉFINITIONS

Exemple lié au schéma statique

Soit la **relation** $R(F_{HDN}, F_{Ac}, F_{Ae})$,
qui lie les trois **flux** F_{HDN}, F_{Ac}, F_{Ae}, et qui s'appuie sur la **méthode** de résolution sous-jacente à la formule : HDN = (A_C - A_e) / A_C . Rappelons que HDN désigne le taux de désazotation de la réaction, A_C l'azote de la charge et A_e l'azote de l'effluent.

Nous décidons que dans cet exemple, cette relation R est **inversible** par rapport aux deux seuls flux F_{HDN}, F_{Ae}; nous supposons que F_{Ac} est une donnée non déductible.

R permet de définir deux **inférences** :
I_{HDN}: F_{Ac}, F_{Ae} -----> F_{HDN}, qui utilise la méthode d'exécution basée sur la formule HDN = (A_C - A_e) / A_C.
I_{Ae} : F_{Ac}, F_{HDN} -----> F_{Ae}, qui utilise la méthode d'exécution capable d'inverser la formule HDN = (A_C - A_e) / A_C.

F_{HDN} et F_{Ae} sont des flux d'**entrée/sortie** de R.
F_{Ac} est un flux d'**entrée** de R.

Les flux F_{HDN} et F_{Ae} sont des informations pertinentes pour l'utilisateur, visibles par lui. Ils correspondent à des **buts** :
B_{HDN} : trouver la valeur de l'HDN,
B_{Ae} : trouver la valeur de l'azote de l'effluent.

Exemple lié au schéma dynamique

Supposons, dans notre exemple, que nous connaissions la valeur de l'azote de la charge A_C et la valeur du taux de désazotation HDN.

On définit la variable V_{HDN} et la variable V_{Ac}, qui ont pour valeur, $valeur_{HDN}$ et $valeur_{Ae}$.

Le **contexte** est alors C = (V_{HDN}, V_{Ac}).

Dans ce contexte, on se pose le problème d'obtenir la valeur de l'azote de l'effluent.
Le **problème** est donné par le doublet P = (C , B_{Ae}).

Pour résoudre ce problème, on procède comme suit : on examine quelles sont les relations qui ont le flux F_{Ae} en sortie. On trouve R. De R, on déduit l'inférence adéquate : I_{Ae}.

Finalement, la **tâche** qui permet de résoudre le problème P est définie par : $T = (C, I_{Ae})$.

V.2.4.2) REPRÉSENTATION STATIQUE DES CONNAISSANCES

Les flux et les relations définis ci-dessus vont donner naissance aux objets ci-dessous.

R de la classe Relation

NOM	R
LIBELLÉ	"Formule de calcul de l'HDN"
FLUX_ES	(F_{HDN}, F_{Ae})
FLUX_ENTRÉE	(F_{Ac})
FORMULE	"HDN = (A_c - A_e) / A_c"

F_{HDN} de la classe Flux

NOM	F_{HDN}
LIBELLÉ	"taux de désazotation"
ALIAS	HDN
BUT	"trouver la valeur de l'HDN"
RELATIONS	(R)
CHEMIN	(Catalyseur, RÉACTION_AZOTE, TAUX)

F_{Ae} de la classe Flux

NOM	F_{Ae}
LIBELLÉ	"azote de l'effluent"
ALIAS	A_e
BUT	"trouver la valeur de l'azote de l'effluent"
RELATIONS	(R)
CHEMIN	(Effluent, AZOTE)

F_{Ac} de la classe Flux

NOM	F_{Ac}
LIBELLÉ	"azote de la charge"
ALIAS	A_c
CHEMIN	(Charge, AZOTE)

Remarquons que les champs BUT et RELATIONS de l'objet F_{Ac} ne sont pas définis, puisque F_{Ac} est en entrée et non en sortie de la relation R.

V.2.4.3) UTILISATION DYNAMIQUE DES CONNAISSANCES

Seuls les attributs définis (non vides) sont indiqués

• étape 0)
objet RésolutioN de la classe Résolution

PROBLÈMES	P
CONTEXTE	(CatalyseuR, ChargE, EffluenT)

P de la classe Problème

BUT	B_{Ae}

• étapes 1 et 2)
objet RésolutioN de la classe Résolution

PB_EN_COURS	P
CONTEXTE	(CatalyseuR, ChargE, EffluenT)
FLUX_INCONNUS	F_{Ae}

P de la classe Problème

BUT	B_{Ae}
TACHES_POSSIBLES	R

• étape 3)
objet RésolutioN de la classe Résolution inchangé
P de la classe Problème

BUT	B_{Ae}
TACHES_EXECUTABLES	R

• étape 4)
objet RésolutioN de la classe Résolution

CONTEXTE	(CatalyseuR, ChargE, EffluenT)

P de la classe Problème

BUT	B_{Ae}
TACHE	R
STATUT	résolu

Quand on recherche la valeur correspondant à l'objet F_{HDN} (par exemple) au cours de la résolution, il faut lancer la méthode VALEUR sur cet objet et le chemin s'interprète comme suit : on cherche l'objet de la classe Catalyseur dans le contexte, on évalue son attribut RÉACTION_AZOTE qui désigne un objet

RéactioN, puis on évalue l'attribut TAUX de l'objet RéactioN trouvé. La méthode VALEUR rend le résultat de la dernière évaluation.

Quand on affecte le résultat trouvé à l'objet , on utilise également le CHEMIN : on cherche l'objet de la classe Effluent dans le contexte de la résolution, puis on affecte le résultat à l'attribut AZOTE.

En 4, il faut lancer la méthode EXECUTER de la relation R avec la liste de paramètres :
((HDN, valeur$_{HDN}$) (A_C, valeur$_{AC}$)). Cette méthode procède comme suit :
- Quels sont les paramètres dont on connaît la valeur ? Réponse : HDN et A_C.
- Quel est le paramètre inconnu ? Réponse : A_e.
- Il s'agit de prendre l'inverse de la formule détenue par R.
- Trouvons cette formule inverse.
- Remplaçons les paramètres connus HDN et A_C par leur valeur.
- Évaluons l'expression obtenue.
- Le résultat de l'évaluation précédente est le résultat de la méthode.

V.2.5) CONCLUSIONS

Quelles sont les parties originales, que nous avons développées et mises en place ? Le découpage en tâches nous a permis de proposer à l'utilisateur des **raisonnements enchaînés**, ou des **utilisations isolées**, ce qui était notre but.

La notion de flux d'entrée-sortie nous a permis de traiter **les tâches de façon inversible**, ce qui nous permet de nous rapprocher de très près de la manière dont les experts raisonnent.

La typologie des relations en une hiérarchie de classes (courbes, programme, formule, etc.) a deux conséquences importantes : la première est que les méthodes d'exécution des tâches sont génériques, et ont seulement besoin d'être réécrites quand on introduit une nouvelle sorte de tâches, c'est-à-dire rarement. Secondement, les utilisateurs peuvent intervenir sur le contenu des tâches, sans remettre en cause la mécanique de résolution. Changer les points de base d'une tâche de classe courbe, n'affecte en rien le déroulement d'une session, si ce n'est que les valeurs déduites sont différentes

C'est ainsi que les méthodes **EXECUTER** et **VALEUR** rendent l'application infiniment modulable et capable de s'adapter à un grand nombre de schémas de raisonnement, sans difficulté majeure.

L'**articulation en problèmes** nous a permis de suivre la résolution facilement. Ceci dit, nous avons développé un chaînage arrière, pour exécuter nos enchaînements. Ce développement a été effectué en utilisant des règles. En fait, cette partie 'moteur' peut très bien être mise en place de façon plus efficace, mais le besoin ne s'en est pas fait sentir pour cette application. Pour y parvenir, l'architecture proposée, en classes Résolution, Problème, Flux et Tâche semblent en tous les cas une très bonne base.

Enfin, il faut mentionner que nous avons donné une **définition aussi rigoureuse que possible** aux mots que nous employons. Ce point est en fait un préalable à toute notre construction. L'étude bibliographique qui va suivre (section V.3) va montrer abondamment comment un vocabulaire flou engendre des difficultés de compréhension autour de la notion de tâche.

V.3) COMPARAISON AVEC LES TÂCHES EN INTELLIGENCE ARTIFICIELLE

Le mot tâche est un mot galvaudé. Il est employé dans des acceptions variables suivant les domaines de l'informatique concernés. Pour justifier notre conception de cette notion, et la façon dont nous l'avons traitée, il faut comprendre quelles sont les différentes idées associées : traitements, activités, inférences, buts, méthodes, plans, problèmes, données, variables, flux, entités, relations, contrôle, stratégie.

Dans le domaine des méthodologies, **SADT** (SADT 89) emploie la notion d'activité, qui renvoie à la notion d'action, de quelque chose à faire et finalement aux tâches. En Intelligence Artificielle, cette notion a été largement développée notamment sous l'appellation des **tâches génériques**, et cette réflexion a été approfondie dans **KADS** (KADS 93).

Du point de vue des architecture basées sur les tâches, nous nous sommes contentés d'une comparaison avec trois applications, DSPL (Brown & 89) inspirée des tâches génériques, SCARP (Willamowski 94b) qui s'appuie sur une architecture d'environnement de problèmes et LISA (Delouis 93), qui illustre la mise en oeuvre immédiate (l'opérationnalisation) du modèle conceptuel.

V.3.1) RÉCAPITULATIF

Dans le tableau suivant, nous avons recherché, à travers les études qui constituent notre base de comparaison avec PRINCE, quelles sont les notions équivalentes aux concepts de relations et de flux, fondamentaux pour nous.

schéma 17 : **équivalence des notions de relation et de flux employés dans PRINCE avec des concepts d'autres études**
légende : TG signifie Tâche Générique, ERP Environnement de Résolution de Problème

étude	relation	flux
SADT	activité	donnée
TG (DSPL)	sous-tâche	
KADS	inférence	rôle ?
ERP (SCARP)	tâche ?	
LISA	méthode	but

La lecture de ce tableau est riche d'enseignements.

• Certaines notions n'ont pas d'équivalence, même approchée. Par exemple, les tâches génériques n'évoquent jamais le concept de flux. D'autres fois, les

équivalences restent approximatives, par exemple, un rôle dans KADS n'est pas exactement un flux dans PRINCE.

• Notre modélisation des raisonnements est basée sur une formalisation objets des **diagrammes de flux de données** (DFD), classiques dans les méthodologies du génie logiciel. Dans ces conditions, tous les travaux (examinés ci-dessus), s'appuyant (entre autres) sur ce type de représentation, vont présenter des ressemblances avec les concepts utilisés dans PRINCE. Ainsi, en est-il de SADT et de KADS, avec des différences cependant sur la notion de rôle, plus abstraite qu'un flux dans KADS.

• Les **différences de vocabulaire** sont frappantes, et introduisent énormément de confusion : pour s'en convaincre, il suffit d'examiner la colonne 'relation' du schéma. Par exemple, la notion équivalente d'une inférence, dans la présentation des tâches génériques de Chandrasekaran, est une sous-tâche, terme censé désigner des tâches élémentaires.

• Les architectures baptisées environnement de résolution de problèmes (ERP) posent les mêmes questions que PRINCE. D'une part, l'architecture informatique est basée sur les tâches, d'autre part une grande attention est portée aux visualisations. Cependant, la modélisation sous-jacente aux tâches est différente, ainsi que l'étude du logiciel SCARP l'illustre. Les concepts proposés par PRINCE sont d'une façon générale **plus simples**; par exemple, dans SCARP, une tâche ne se comprend qu'en analysant les concepts associés de stratégie, de sous-tâches, de contexte et de méthode.

• LISA, l'architecture construite par I. Delouis, isole les concepts de méthode et de but. Une méthode dans LISA est une inférence dans PRINCE, et un but dans LISA est un but dans PRINCE. Cependant, et c'est la grande différence, importante sur le plan pratique de la réalisation informatique, Delouis ne distingue qu'**une seule classe de méthode** et qu'**une seule classe de but**. Les différences de comportement, inévitables entre deux objets méthode ou but, sont reportées sur l'interprétation de mots clefs particuliers, présents dans les attributs.

V.3.2) TÂCHES ET MÉTHODOLOGIES

V.3.2.1) TÂCHES ET SADT

Les méthodologies du génie logiciel mentionnent trois axes de description des systèmes informatiques : les données, les fonctions et les états. Ce qui nous importe ici, est de présenter ce que peuvent être des fonctions, ou des traitements. Deux approches nous semblent particulièrement adaptées dans ce but : les méthodes de programmation structurées et SADT. Supposant la programmation structurée largement connue, nous ne parlerons que de SADT.

SADT s'inspire à l'origine de la pensée systémique, et cette méthode est représentative de l'analyse structurée. Son nom signifie 'structured analysis design technic'. Le point d'attaque dans SADT est le recueil des traitements.

SADT est basée sur des schémas de systèmes, qui se décomposent en sous-systèmes. Dans SADT, comme dans les méthodes de tendance systémique, les raisonnements sont vus comme des boîtes qui permettent de transformer des entrées en sorties. Ces boîtes sont nommées **activités**, et elles agissent sur des données d'entrée ou de sortie. En décomposant ces boîtes, on tombe de façon classique sur les tâches élémentaires, c'est-à-dire celles qu'on ne souhaite pas décomposer plus. Comme on scinde les informations en données et traitements d'une manière tout à fait habituelle, on retrouve des schémas dits *datagrammes*, qui permettent d'examiner les données, inverses des *actigrammes* qui se focalisent sur les traitements. SADT a été largement inspiré le projet SAGACE (Penalva 1990), à EDF.

Comme on enchaîne des données, aucune structuration n'est supposée pour ces dernières. Libre à nous de les représenter sous forme d'objets. De plus, SADT ne précise pas ce que peuvent être données d'entrée, de sortie ou de contrôle, et il peut être difficile de faire la distinction entre ce qui est entrée ou contrôle par exemple.

Dans un actigramme, un traitement va faire le lien entre des données d'entrée et de sortie. Une activité sera donc un opérateur qui transforme des données d'entrée en données de sortie. Cet opérateur sera identifié par un nom et une fonction (qui explicite son but). La méthode employée pour faire cette transformation ne sera pas décrite à ce niveau, elle reste implicite, mais elle est supposée connue.

SADT offre une représentation agréable, pour expliquer comment des pas de raisonnements (nommés activités) s'enchaînent. Si la problématique des tâches, c'est de faire des plans, SADT nous a semblé un premier outil. Concrètement, une activité va devenir une inférence pour nous.

V.3.2.2) LES TÂCHES GÉNÉRIQUES SELON CHANDRASEKARAN

Le concept de tâche générique ne fonde pas, à proprement parlé, une méthodologie, mais il a été repris et développé dans KADS, que l'on présente ensuite. Cette courte revue non exhaustive s'inspire d'une note du laboratoire d'informatique fondamentale de Lille (Dekker, 92). Chandrasekaran a publié énormément d'articles qui traitent de cette question. Sa vision des choses s'est affinée et a évolué, ce qui rend une synthèse d'autant plus délicate.

La notion de tâche générique part de la constatation qu'il n'y a **pas de manière universelle** de résoudre un problème, mais seulement des façons spécifiques : les problèmes se différencient et se scindent en plusieurs grandes classes. Chaque fois qu'on aura identifié une classe de problèmes, les types de résolution valables pour cette classe seront appelés des tâches, et ces tâches seront génériques parce

que s'appliquant à une **classe de problèmes** plutôt qu'à un problème particulier. Ainsi, en est-il de la tâche de diagnostic.

Ces tâches utiliseront des connaissances particulières, et feront l'objet d'un contrôle (c'est-à-dire d'un mode de raisonnement) spécifique. Elles se décomposent pour former des sous-tâches, jusqu'au niveau le plus fin où elles peuvent s'exécuter, par l'intermédiaire de méthode(s) (Chandrasekaran & 92).

Après avoir explicité ces tâches, on peut construire des architectures adaptées, capables de les manipuler. On peut citer ainsi DSPL (Brown & 89), qui s'applique à des problèmes de conception de routine comme un exemple de générateur construit selon cette optique.

Ceci dit, 'les mêmes tâches peuvent surgir dans des processus différents de résolution de problèmes'. Autrement dit, un problème complexe, tel que le diagnostic, peut se décomposer en plusieurs types de raisonnement (classification, abduction, inférence), qui font eux l'objet de tâches génériques, et qui peuvent se retrouver dans d'autres classes de problèmes (la conception, par exemple).

En conception, Chandrasekaran, dans un article paru dans AI magazine (Chandrasekaran, 90) propose une stratégie globale, qui s'énonce par 'Propose Verify Critic and Modify' (PCM), où le V de 'verify' est curieusement omis. On propose une solution. On l'évalue ou on la vérifie. Si cette proposition ne convient pas, on la critique et on la modifie. On itère jusqu'au succès ou jusqu'à l'échec de cette hypothèse. En cas d'insuccès, on essaie une autre solution.

Quelles sont les méthodes, capables de remplir les fonctions de proposition. Chandrasekaran énumère : des méthodes de décomposition, des méthodes à base de cas, des méthodes de satisfaction de contraintes, des méthodes d'optimisation numériques, des inférences. Quand il s'agit de vérifier, on trouve des calculs spécifiques, et des méthodes de calculs qualitatifs entre autres. On pourrait passer en revue l'ensemble de chacun des items de l'analyse P(V)CM. On remarquerait que beaucoup de techniques, appartenant ou non au domaine de l'intelligence artificielle, trouvent un emploi à un moment ou à un autre de la démarche.

Chandrasekaran conclut cet article, en remarquant que pour informatiser les problèmes de conception, le problème clé est de définir les liens entre tâches, sous-tâches, méthodes et domaine.

En fait, montrer comment tâches, sous-tâches, méthodes et domaine sont reliés, est le problème central, et il est curieux qu'ayant correctement identifié la principale difficulté, Chandrasekaran ne propose pas d'architecture capable de représenter les tâches. Le concept de tâche générique est encore aujourd'hui **peu opérationnel**.

Au contraire, mettre en oeuvre une architecture basée sur les tâches est ce que nous avons essayé de réaliser dans PRINCE.

V.3.2.3) KADS

Nos références sont tirées de (KADS 93)

KADS est une méthodologie développée dans le cadre des programmes ESPRIT, et un nombre important de sociétés de service informatique et de laboratoires universitaires européens y sont associés. Son but est de créer une formalisation de la construction des systèmes à base de connaissances, qui s'impose comme une norme, notamment aux USA, où le marché potentiel reste très substantiel.

V.3.2.3.1) présentation

KADS veut produire des modèles dont on puisse réutiliser certaines parties. Pour ce faire, cette méthodologie décrit à la fois finement comment accompagner la conception informatique d'une application, et aussi quels sont les différents modèles qu'il faut finalement produire.

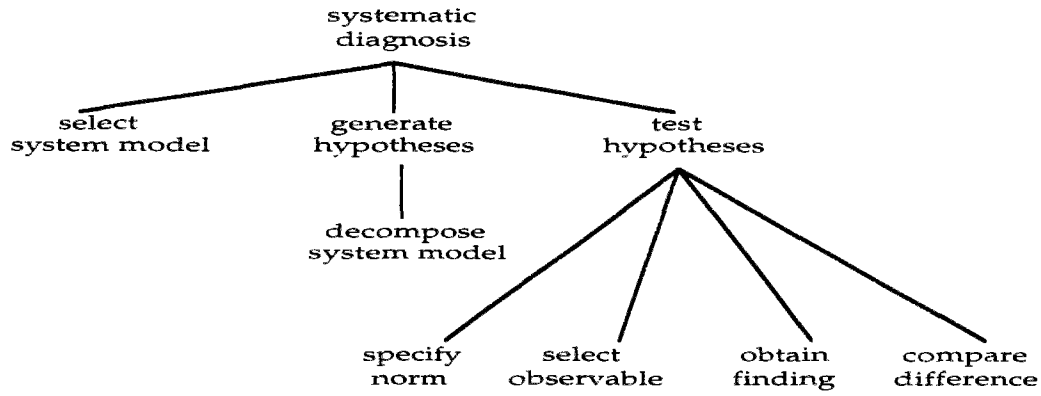
KADS distingue quatre niveaux de représentation, les niveaux stratégie, tâches, inférence, domaine.

Le niveau stratégie doit permettre de décider quelle est la méthode à employer pour résoudre chaque tâche. Par exemple, si le raisonnement conduit à une impasse, il doit être possible de trouver une nouvelle approche. On s'occupe ici surtout d'exécution des tâches et d'examen des échecs.

Le niveau tâche indique comment les inférences doivent s'enchaîner pour arriver à un certain but. Une tâche peut donc se décomposer en inférences (des tâches élémentaires), en sous-tâches, et en tâches dites de transfert, c'est-à-dire en interactions avec l'utilisateur ou avec d'autres systèmes (réception d'informations de la part de capteurs par exemple). En fait, le niveau tâche a pour ambition de détailler quel est le contrôle exercé sur le niveau inférieur des inférences, pour ne pas retomber dans les erreurs d'un système tel que MYCIN (Clancey 92) où règles d'expertise et contrôle du raisonnement se mélangent sans qu'on puisse très bien faire la distinction.

Voici l'exemple d'un découpage en tâche, tiré de la référence (KADS 93), d'une application destinée à diagnostiquer les défauts d'un système auditif.

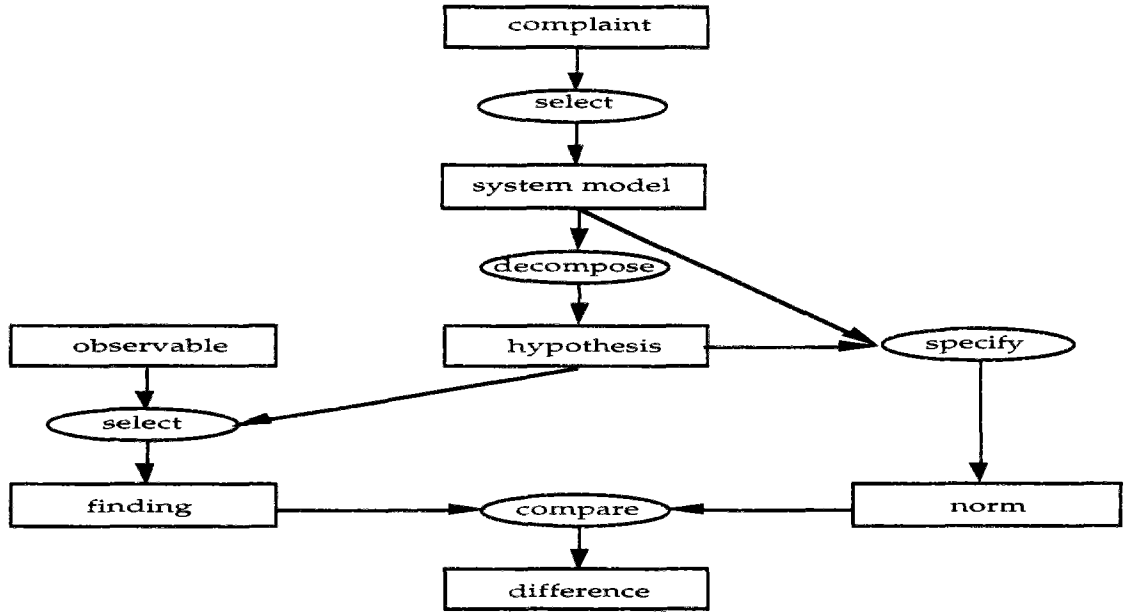
schéma 18 : **exemple de découpage en tâches dans KADS**



Le niveau inférence décrit les agents de résolution, ce que KADS nomme les sources de connaissance, qui comprennent un algorithme (c'est-à-dire un moyen d'aboutir au résultat), des informations d'entrée et de sortie, et une caractérisation de la transformation effectuée. Ici, il s'agit de détailler les inférences potentielles, c'est-à-dire les raisonnements élémentaires qui peuvent être conduits. On ne choisit pas à ce niveau parmi les inférences candidates, on ne dit pas non plus dans quel ordre elles doivent être menées.

Comme KADS veut construire des méthodes réutilisables, on ne dit pas non plus au niveau des inférences (caractéristiques du comment), le pourquoi des actions ni leur but. Seules les tâches disposent de cette information, et le niveau inférences sera de cette façon le moins enclin à changer d'une application à une autre.

schéma 19 : **exemple de schéma d'inférences dans KADS** (même référence)



Le niveau domaine est la représentation (objet) du domaine d'application.

V.3.2.3.2) discussion sur KADS

Ce qu'il faut remarquer, c'est que KADS se situe d'emblée à un haut niveau d'abstraction. La preuve en est que pour formaliser toute cette présentation, un langage mathématique spécifique a été construit nommé $(ML)^2$ (voir KADS 93).

Il faut noter également que ce vocabulaire et cette représentation diffèrent sensiblement de ce que la littérature nous présente habituellement. Si l'identification du niveau domaine est une démarche naturelle, il n'en va pas de même des trois autres couches. Ce qu'il faut bien comprendre, c'est que KADS essaie avant tout de définir un cadre structuré qui permette de raisonner sur les tâches. Ce qu'il faut éviter, c'est de représenter le contrôle du raisonnement sous forme par exemple de méta-règles comme dans Neomycin, c'est-à-dire d'un savoir qui ne se distingue en rien des règles expertes. On veut pouvoir changer également de ligne de raisonnement, par exemple passer d'un raisonnement par hypothèses à un autre style de raisonnement, et on s'oblige du coup, à caractériser chaque type d'inférence.

En fait, tâche et contrôle (du raisonnement) ont même signification dans KADS, et on peut se demander pourquoi ce n'est pas le mot, plus clair, de contrôle qui est employé, plutôt que celui de tâche. D'autant plus que ce contrôle (sur les inférences) ne diffère en rien de ce que l'on pratique depuis toujours dans les langages algorithmiques : les inférences vont s'enchaîner à travers des séquences (*a then b*), des tests (*if f then a else b*), des boucles (*repeat a until b*) (voir $(ML)^2$.

Il reste que cette idée de représentation explicite du contrôle du raisonnement est très intéressante. Elle oblige à scinder le raisonnement en étapes élémentaires, véritables briques de base, sur lesquelles l'application va se construire. Elle permet d'utiliser un langage déclaratif au niveau des inférences, plus proche de la façon de s'exprimer des experts. Enfin, le contrôle lui-même peut devenir un objet d'analyse et de raisonnement.

Au niveau des inférences, le langage particulier de KADS masque qu'on essaie de bâtir un **diagramme de flux de données**. A la nuance près, que KADS introduit une taxinomie, une classification de ces modèles d'inférence, pour qu'on puisse les réutiliser. On devrait par exemple pouvoir repartir d'un diagramme général d'inférences sur le diagnostic, qu'on adapterait à son problème particulier. Encore faut-il que l'expérience confirme ce point.

Réfléchir aux inférences, c'est tracer un diagramme de flux de données. Et comme nous l'avons vu, une inférence (KADS), un enchaînement représenté par un diagramme de flux de données, ou encore une activité (SADT) ont une représentation équivalente.

Enfin, du point de vue des tâches, KADS suppose a priori que le découpage est fixe : l'exemple précédent d'un diagramme de tâches est très parlant de ce point de vue.

V.3.2.4) SYNTHÈSE

Comme nous l'avons déjà mentionné, nous sommes repartis d'une représentation des inférences à la façon de SADT, qui peuvent s'assimiler en grossière approximation à des actigrammes.

Dans les tâches génériques (TG), KADS, le terme tâche est réservé au contrôle du raisonnement, c'est-à-dire au schéma d'enchaînement des buts. Quand on décrit la façon dont se suivent les différentes étapes du raisonnement, on évoque des tâches; quand on cherche à décrire la façon dont est accomplie chaque étape élémentaire, on parle de sous-tâche et d'inférence.

Nous avons soutenu pour notre part que la nature élémentaire ou composée d'une étape de raisonnement ne change rien à notre façon de la considérer, et c'est également le point de vue d'un utilisateur. Seule importe la méthode de résolution, qui guide le niveau de décomposition et d'analyse requis.

Enfin, la hiérarchie de classes que nous proposons pour représenter les différents types de relation, conserve le caractère explicite du contrôle du raisonnement, point fondamental des méthodologies KADS et TG. L'examen des relations permet selon leur classe de connaître immédiatement le type de contrôle qu'il faut exercer pour exploiter les connaissances détenues, comme on peut le faire dans KADS par l'analyse des tâches.

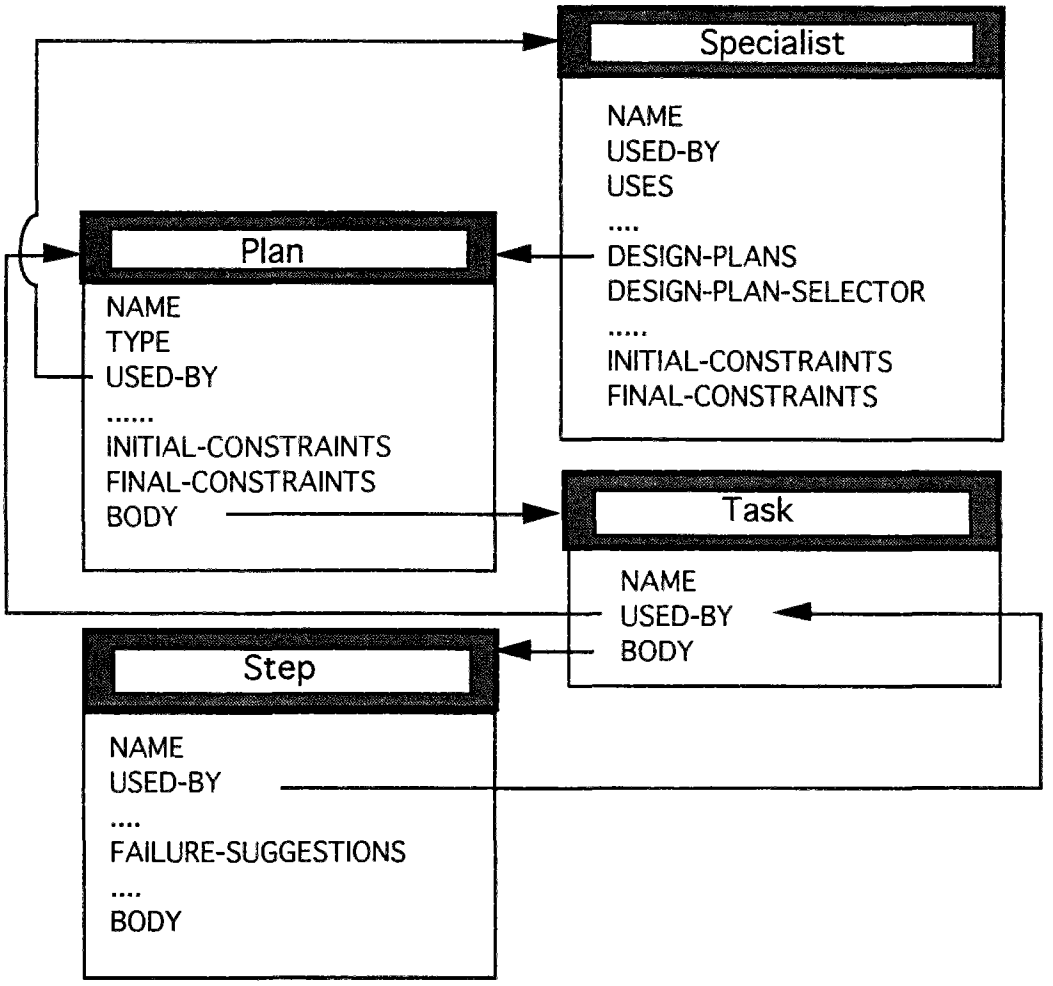
V.3.3) QUELQUES TRAVAUX EN IA SUR LES TÂCHES

Nous avons choisi pour finir cette présentation de la notion de tâches de décrire trois applications. La première illustre le point de vue de Chandrasekaran. Il s'agit de DSPL . Les deux suivantes, SCARP et LISA illustrent mieux la direction vers laquelle il faut progresser, selon notre point de vue.

V.3.3.1) DSPL : 'DESIGN STRUCTURE AND PLANS LANGUAGE'

DSPL est présenté comme un langage pour les systèmes experts de conception et s'appuie sur l'analyse des tâches génériques de Chandrasekaran. La conception est vue ici comme la faculté de construire un plan des actions à mener pour concevoir un certain artefact. DSPL a été en particulier utilisé pour la conception d'un dispositif de compression à air - 'air cylinder'-, mais aussi pour la planification de missions d'avions (Chandrasekaran & 89).

schéma 20 : **hiérarchie des concepts dans DSPL**



Détail de la classe 'Specialist'
NAME (nom)
USED-BY et USES (utilisé par et utilise). La résolution se décrit par l'intermédiaire d'une hiérarchie de spécialistes. Ainsi, un spécialiste connaît les spécialistes qu'il utilise (attribut 'uses') et ceux dont il dépend (attribut 'used-by').
INITIAL-CONSTRAINTS, FINAL-CONSTRAINTS (contraintes initiales et finales). Elles correspondent aux pré et post conditions.

Détail de la classe 'Plan'
NAME, USED-BY, INITIAL/FINAL-CONSTRAINTS : même chose que pour les spécialistes.
BODY (corps) : c'est une liste de tâches (de la classe 'task') que le plan doit exécuer. En fait, un plan est une sorte de spécialiste dont chaque item doit être exécuté séquentiellement.

Détail de la classe 'Task'
BODY (corps) : cet attribut désigne ici une liste d'étapes élémentaires (classe 'step').

Détail de la classe 'Step'
BODY (corps) : liste d'actions élémentaires.
FAILURE-SUGGESTIONS (suggestion en cas d'échec) : si une étape échoue, DSPL dispose de toute une panoplie d'actions de remplacement et de corrections, pour se sortir de cette mauvaise situation. Cet attribut 'failure-suggestions' détient une liste d'hypothèses de modifications et il est complété par d'autres actions dont nous n'entreprenons pas la description ici, qui permettent de repartir de telle ou telle étape du cheminement.

La principale critique que l'on peut adresser, à notre avis à DSPL, est que les concepts utilisés (plans, tâches, items, spécialistes, etc.) sont flous, et qu'il est difficile de les distinguer les uns des autres, et donc ardu de les mettre en oeuvre. En second lieu, la décomposition des problèmes est supposé fixe.

Ceci dit, ce type de remarques pourraient être adressées à d'autres applications,, notamment PRIDE (Mittal & 86) ou encore NMS (Corby 91), quoique la vocation de cette dernière application soit légèrement différente.

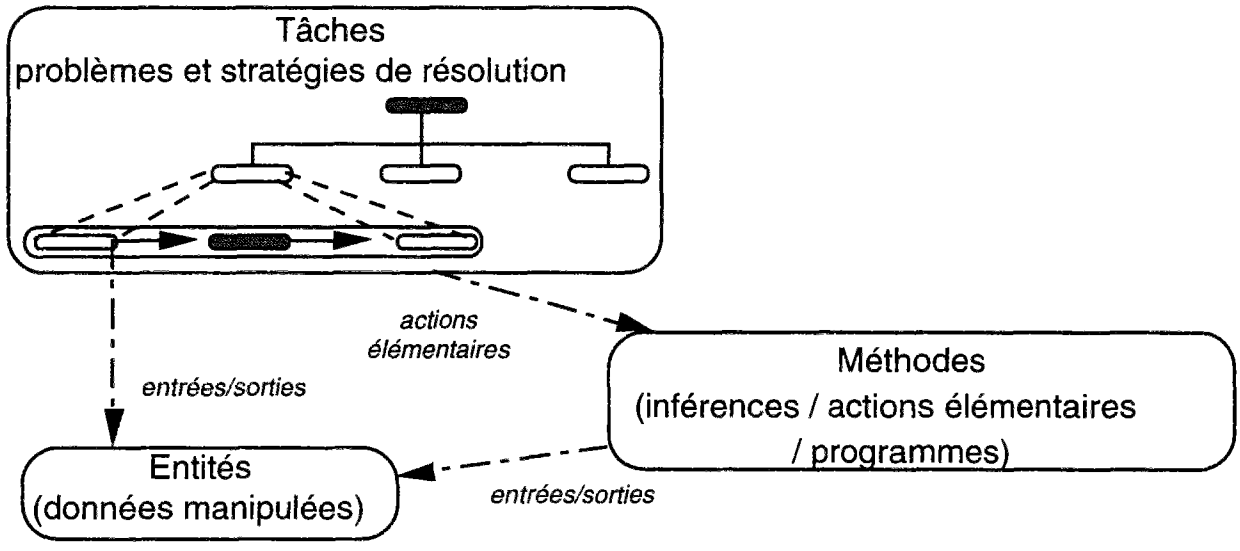
V.3.3.2) SCARP, TÂCHES ET RÉSOLUTION DE PROBLÈMES

L'approche des environnements de résolution de problème (ERP) "repose sur une base de connaissances, qui permet d'intégrer sous forme déclarative des méthodes (calcul, graphique), les objets impliqués dans ces méthodes (tableaux, équations modèles) et l'expertise relative au choix, à l'utilisation pratique et à l'interprétation des résultats des méthodes" (Rousseau &, 1989). Cette approche nous parait particulièrement intéressante, et c'est **le même point de vue** que nous avons adopté dans PRINCE.

Bertrand Rousseau (ROUSSEAU 1988) décrit une tâche comme une interface entièrement déclarative qui encapsule la méthode qui la réalise. Une tâche contient d'une part les fonctionnalités de la méthode, c'est-à-dire son but, ses conditions d'emploi et ses effets, et d'autre part son fonctionnement, c'est-à-dire la façon de la mener à bien. Cette définition renvoie encore une fois à des notions dont nous avons fait usage dans PRINCE.

Les travaux que Rousseau a initié se continuent aujourd'hui avec notamment Willamowski (Willamowski 94a et b), qui a construit le générateur SCARP. Une base de connaissances SCARP est définie à partir de tâches, de méthodes et d'entités, comme le précise le schéma ci-dessous, tiré de (Willamowski 94b).

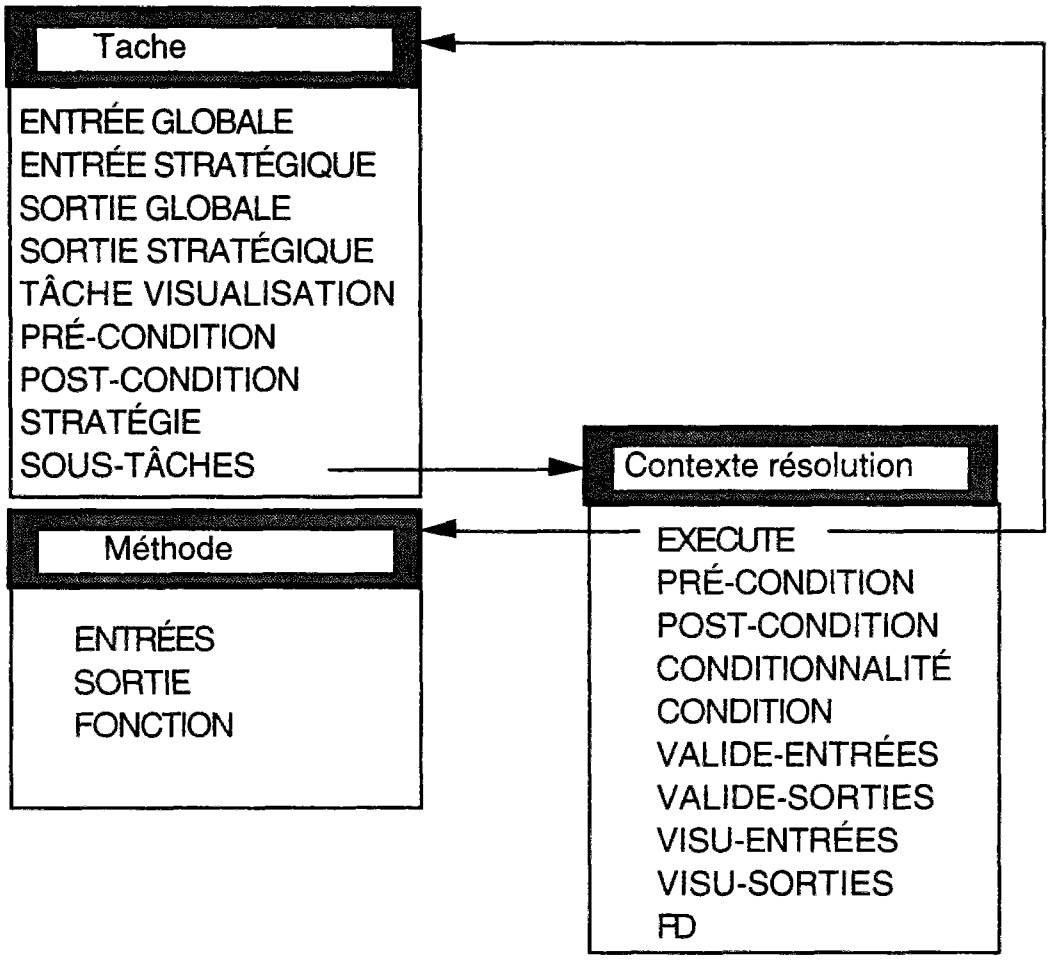
schéma 21 : **tâches, méthodes et entités dans SCARP**



Willamowski a construit SCARP, dont le nom signifie Système Coopératif d'Aide à la Résolution de Problèmes. Cette architecture, opérationnelle, est construite pour que le système et l'utilisateur puisse coopérer, et ce dernier peut même au cours d'une session, indiquer au système quel raisonnement suivre.

Cette architecture s'appuie sur la modélisation suivante :

schéma n°22 : **modélisation des tâches dans SCARP**



Discussion sur la classe Tâche de Willamowski

ENTRÉE/SORTIE, GLOBALE/STRATÉGIQUE : une entrée globale est transformée en une entrée stratégique par l'intermédiaire d'un pré-traitement, appelé pré-tâche. Une entrée (respectivement sortie) globale se présente sous une forme que l'utilisateur connaît , une entrée (resp. sortie) stratégique est adaptée à la tâche qui va suivre. Pourquoi ce pré-traitement sur les entrées et ce post-traitement sur les sorties ne font pas l'objet d'une tâche séparée, clairement identifiée ?

TÂCHES DE VISUALISATION DES ENTRÉES OU DES SORTIES : de même qu'un traitement préalable est défini sur les entrées (respectivement les sorties) avec l'attribut entrée (sortie) globale, on peut associer aux entrées (sorties) une tâche de visualisation, activable à la demande. De même que pour les entrées globales ou stratégiques, pourquoi une visualisation ne fait pas l'objet d'une tâche séparée ?

PRÉ, POST-CONDITIONS : il s'agit de conditions portant sur les entrées et les sorties d'une tâche, qui en restreignent l'utilisation et la validité.

STRATÉGIE : cet attribut permet de préciser le contrôle (à l'aide de mots clefs) exercé sur les sous-tâches. Outre les opérateurs classiques *séquence, parallèle,*

choix et *itération*, nous avons noté les contrôles *choix-utilisateur* et *utilisateur*, qui demandent à l'utilisateur d'intervenir dans le déroulement de la tâche, ce qui permet de gérer une certaine coopération entre l'homme et la machine. Dans PRINCE, tous ces types de contrôle différent auraient donné naissance à autant de classes 'relation', munies à chaque fois des méthodes d'exécution adéquates.

SOUS-TÂCHES : Les (sous)-tâches contrôlées par la tâche sont mentionnées ici. Cet attribut vient en complément de l'attribut stratégie.

Discussion sur la classe Contexte de résolution de Willamowski
Le contexte d'exécution sert à décrire comment les sous-tâches sont liées à la tâche principale.

EXECUTE : 'exécute' référence la méthode d'exécution de la sous-tâche. Quand il s'agit d'une sous-tâche complexe, elle donne lieu à un second niveau de décomposition et dans ce cas, 'exécute' renvoie sur une tâche.

CONDITIONNALITÉ et CONDITION : ces deux attributs permettent de préciser si la réalisation de la tâche est obligatoire, optionnelle ou conditionnelle et dans ce cas sous quelles conditions.

PRÉ, POST-CONDITIONS : les pré-conditions (et les post-conditions) d'exécution sont ici renforcées, et précisées à nouveau. A notre avis, ces attributs qui touchent aux conditions d'exécution (conditionnalité, ..., post-conditions) introduisent des distinctions fines, peut-être difficiles à manipuler.

VALIDE-ENTRÉES, VALIDE-SORTIES, VISU-ENTRÉES, VISU-SORTIES : des traitements préalables ou postérieurs de validation par l'utilisateur ou de visualisation sont ici précisés. Pourquoi ne font-ils pas l'objet de tâches séparées ?

FD : fd signifie flot de données. Cet attribut précise les flux de valeurs entre les différentes sous-tâches référencées par la tâche principale. Ces flots de données sont décrits dans PRINCE par l'intermédiaire de la classe 'flux'.

Description de la classe méthode
Nous avons rassemblé sous cette rubrique les classes 'méthode-interne' et 'méthode-externe' de SCARP. La description que nous en donnons est à rapprocher de la classe 'Programme' de PRINCE.

ENTRÉES, SORTIE : type des données d'entrée et de sortie
FONCTION : nom de la fonction sous-jacente.

SCARP et PRINCE présentent quelques différences importantes.
- La notion de but n'est pas apparente dans SCARP.
- PRINCE n'utilise pas la notion de sous-tâche.

• Les différents types de méthode se décrivent par l'intermédiaire de différentes classes 'Relation' (issues d'une même racine) dans PRINCE, alors que toutes les possibilités d'une structuration 'objet' ne sont pas utilisées dans SCARP.
• Le flot de données est une des notions principales de PRINCE, alors que ce n'est qu'un attribut du contexte de la classe 'contexte d'utilisation' dans SCARP.
• Les notions entrées-globales, sorties-globales, valide-entrées, valide-sorties, visu-entrées, visu-sorties donneraient lieu dans PRINCE à des tâches distinctes.

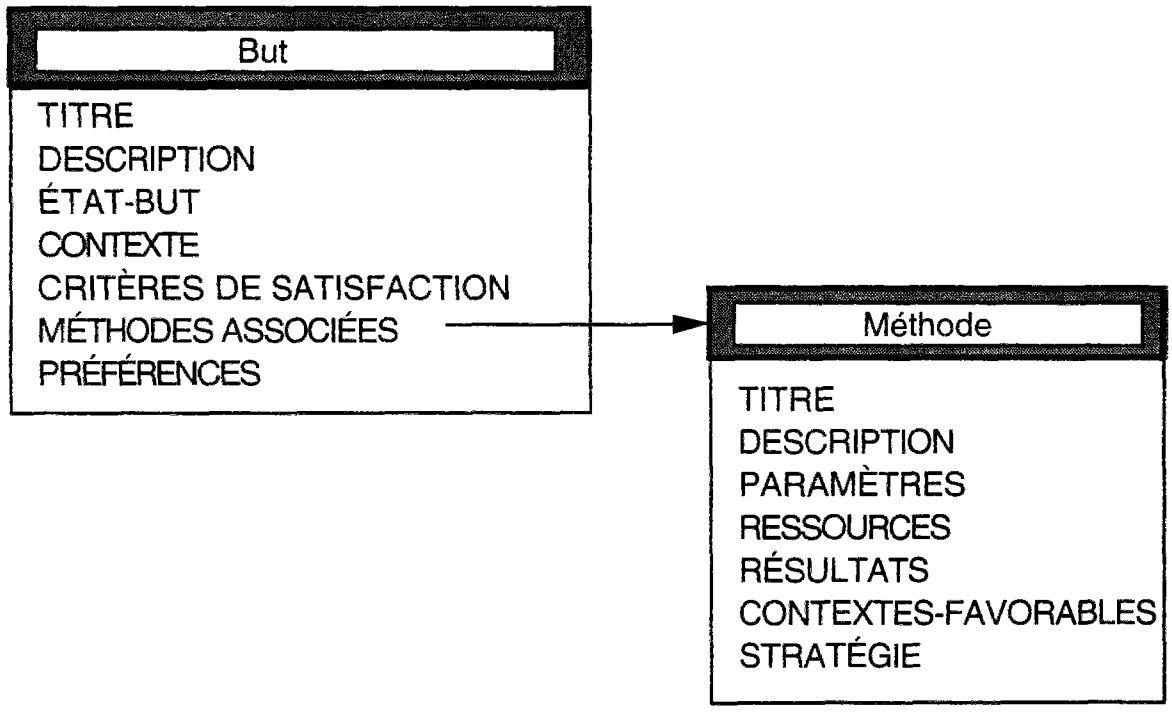
V.3.3.3) LISA : OPÉRATIONNALISATION DU MODÈLE CONCEPTUEL

Isabelle Delouis (Delouis 93) décrit une architecture destinée à rendre directement opérationnelle, c'est-à-dire à la disposition de l'utilisateur, le modèle conceptuel mené.

C'est la bonne direction, car autant il est facile, l'expérience le prouve, de discuter du modèle conceptuel avec les utilisateurs ('knowledge level'), autant l'architecture informatique mise en oeuvre ('symbol level') reste souvent absconse pour un non-informaticien, et parfois abstruse, même pour un public averti. Autant donc sauter cette dernière étape et passer directement de la conception à l'implémentation informatique.

Cette idée de sauter l'étape d'analyse purement informatique, de passer directement de la conception à l'implémentation est d'ailleurs inscrite dans une méthodologie orientée objet, celle de Schlaer et Mellor (Schlaer & 88).

schéma 23 : **modèle proposé par Isabelle Delouis**



Discussion sur la classe But de Delouis

Passons sur les champs TITRE et DESCRIPTION, correspondant à NOM et LIBELLE de la classe But dans PRINCE.(voir le paragraphe V.2.2.2)

ETAT-BUT : ce champ désigne en fait le type de résultat à obtenir, et des contraintes caractérisant les propriétés de l'état but.

- Pourquoi ce mélange d'une description avec des contraintes ?
- Par ailleurs, la façon de décrire un ETAT-BUT suit une grammaire particulière. Par exemple : réseau-à-étudier (un réseau) fait référence à un objet de la classe Réseau, la classe Réseau faisant partie de la description du domaine, l'application cible de Delouis étant une étude de réseau électrique à EDF. Dans PRINCE, cette sémantique est remplacée par une hiérarchie de classes suivant le type de flux.

CONTEXTE : cet attribut définit les éléments devant être disponibles pour que le but ait un sens.

- S'agit-il de valeurs d'entrée, de contraintes, d'éléments de contrôle ? Le contexte est-il lié au but ou aux méthodes ?

CRITÈRES DE SATISFACTION : critères permettant d'évaluer la qualité des solutions obtenues.

- On souhaite répondre à la question : est-il souhaitable de déclencher une autre méthode ? Les critères de satisfaction sont de ce point de vue un élément de contrôle sur le déroulement du raisonnement.

MÉTHODES ASSOCIÉES : méthodes connues par les experts, pour atteindre le but.

- Mais, dans un problème réel, existe-t-il des méthodes inédites qu'un système à base de connaissances serait capable de découvrir de façon autonome ?

PRÉFÉRENCES : connaissances permettant d'établir des priorités entre les différentes méthodes

- Comme dans le cas des critères de satisfaction, cet attribut définit un élément de contrôle sur le raisonnement.
- Les attributs critères de satisfaction et préférences sont à rapprocher. On préférera a priori les méthodes qui risquent de donner satisfaction.

Discussion sur la classe Méthode de Delouis

PARAMÈTRES : description des paramètres d'entrée de la méthode et de contraintes sur ces paramètres.

- Mêmes remarques que pour les états-buts.

RESSOURCES : ressources nécessaires à la mise en oeuvre de la méthode.

- Voir l'attribut stratégies

RÉSULTATS : résultats produits par la méthode.

- Il s'agit donc des sorties. Dans PRINCE, un but correspond à la sortie d'une inférence (appelée ici méthode). De ce point de vue, il y a un problème de redondance, et peut-être de cohérence avec la notion d'état-but.
- Mêmes remarques que pour les états-buts.

CONTEXTES FAVORABLES : description des conditions favorables à l'utilisation de la méthode

- Il peut être délicat de faire la distinction entre l'attribut contexte situé au niveau des méthodes, avec celui porté par le but.

STRATÉGIES : pointeur vers l'entité opératoire : code de calcul, base de règles, plan de résolution

- De façon évidente, ressources et stratégies se complètent, pour définir la méthode de résolution. Pour distinguer les diverses ressources, et les diverses stratégies, des mots clefs suivis de qualificatifs adéquats sont nécessaires, alors que dans PRINCE, cette description se fait par une distinction en classes.

Si une tâche doit se décomposer, alors cette décomposition est indiquée dans le corps d'une des méthodes associées. C'est pourquoi I. Delouis distingue trois classes de méthode :
- les méthodes terminales, qui ne peuvent être qu'exécutées; on peut citer dans le cas de cet auteur, des codes de calcul, ou des procédures.
- les décompositions en étape, qui permettent d'enchaîner des tâches, et qui utilisent les connecteurs TANT QUE et SI...SINON SI. Notons l'absence pour le moins étonnante du connecteur ET. En fait, comme on veut se donner la possibilité de raisonner en termes de buts, ce ne sont pas des tâches qui sont décrites dans cette décomposition, mais des buts. La suite exacte de méthodes employées sera finalement déterminée au moment de l'exécution.
- l'utilisateur, qui est pour chaque tâche à traiter, une des méthodes applicables. Comme le note I. Delouis, cela n'a pas toujours un sens.

Les différences principales avec PRINCE, pour résumer, sont les suivantes :
- Les différents types de méthode se décrivent par l'intermédiaire de différentes classes Inférences (issues d'une même racine) dans PRINCE, alors que toutes les possibilités d'une structuration 'objet' ne sont pas utilisées dans LISA
- Les différents types de résultats et données d'entrée se décrivent par l'intermédiaire de différentes classes Flux (issues d'une même racine) dans PRINCE
- Les contraintes d'utilisation d'une méthode dans LISA sont réparties entre les objets buts et les méthodes.

V.4) CONCLUSIONS

Après avoir analysé la façon dont les experts s'y prennent, nous avons successivement défini les mots tâches, inférences, buts, méthodes, ce qui nous a permis de poser un premier schéma de représentation des connaissances. L'examen plus poussé des buts nous a amené à les considérer comme un type particulier de flux (flux de sortie)

Nous avons alors montré comment ce schéma était utilisé dans PRINCE, notamment comment les relations inversibles étaient mises à disposition des experts, et comment les méthodes EXECUTER pour les relations et VALEUR pour les flux permettaient de s'adapter à beaucoup de schémas de raisonnement.

Nous avons décrit notre technique de résolution, en insistant moins sur son caractère idéal, que sur la représentation des raisonnements qu'elle emploie, et sur la notion de problèmes.

Enfin, nous avons comparé notre travail avec d'autres architectures et d'autres réalisations basées sur les tâches. Parmi les méthodologies examinées, SADT offre pour nous une première base avec les actigrammes. L'abandon de la notion de sous-tâche et la pleine utilisation des possibilités de représentation objet conduisent finalement à des différences importantes avec les tâches génériques et surtout avec KADS.

Du point de vue des applications, DSPL nous semble peu aisé d'emploi, essentiellement parce que les concepts utilisés sont difficiles à comprendre. SCARP est une architecture opérationnelle, qui s'adresse comme PRINCE à un public d'ingénieurs pour résoudre des problèmes. Cependant, l'orientation poussée de cette architecture vers la coopération homme-machine engendre un ensemble de notions que PRINCE n'utilise pas. LISA, enfin, utilise un langage là où PRINCE s'appuie sur les possibilités des langages à objets.

V.A) ANNEXE

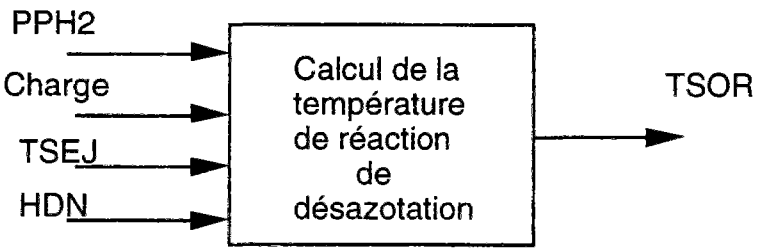
V.A.1) POINTS DE DISCUSSION

Plusieurs points sont à souligner :

• 1) Notre démarche par chaînage arrière n'est possible, que parce que l'arbre des buts, dans les parties principales du raisonnement de PRINCE, peut s'exprimer sous la forme d'un **graphe sans circuit**. Les retours arrière sont actuellement gérés par les experts qui apprécient si il y a lieu de recommencer les raisonnements.

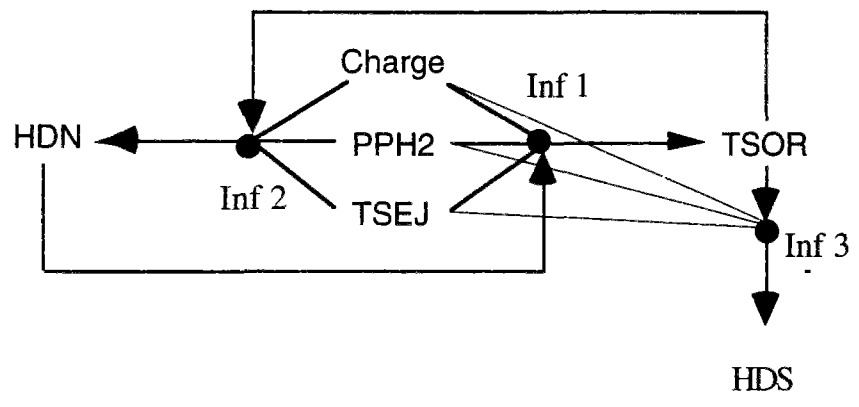
• 2) Si l'on néglige le succès ou l'échec des actions qui sont lancées, on doit pouvoir se raccrocher à un formalisme d'ordre 1 strictement. L'échec d'une inférence peut malheureusement arriver.
Par exemple :
A partir de la donnée d'une charge, et d'un objectif souhaité de niveau d'azote dans l'effluent (HDN), de la connaissance de la pression(PPH2) et du temps de séjour (TSEJ), on peut calculer la température (TSOR) nécessaire :

schéma 24 : **inférence '*calcul de la température de désazotation*'**



Mais il arrive que la température trouvée soit trop élevée (supérieure à TSOR-limite, cette dernière température étant fixée par les experts). Dans ce cas, on impose la température fixée à TSOR-limite, et on recalcule l'HDN que l'on obtient, toutes les autres conditions restant identiques.

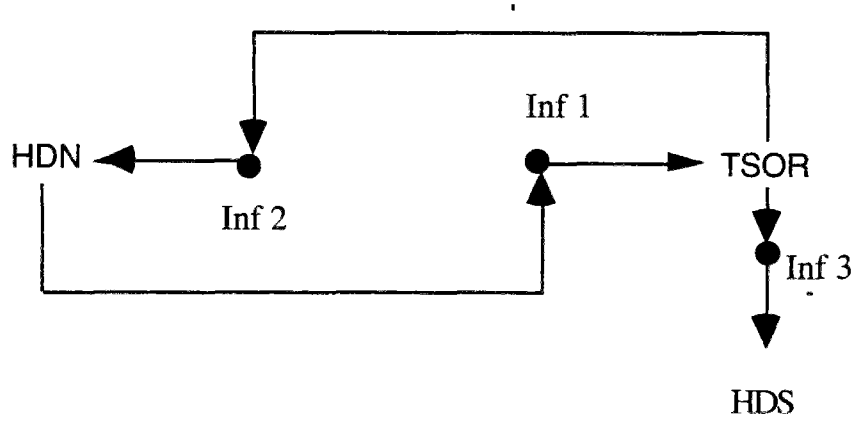
schéma 25 : **graphe d'enchaînement des inférences**



1) HDN, Charge, PPH2, TSEJ donnent (inférence 1, notée Inf 1 sur le schéma) TSOR.
2) TSOR, Charge, PPH2, TSEJ donnent (inférence 2) HDN.
3) TSOR, Charge, PPH2, TSEJ donnent (inférence 3) HDS.

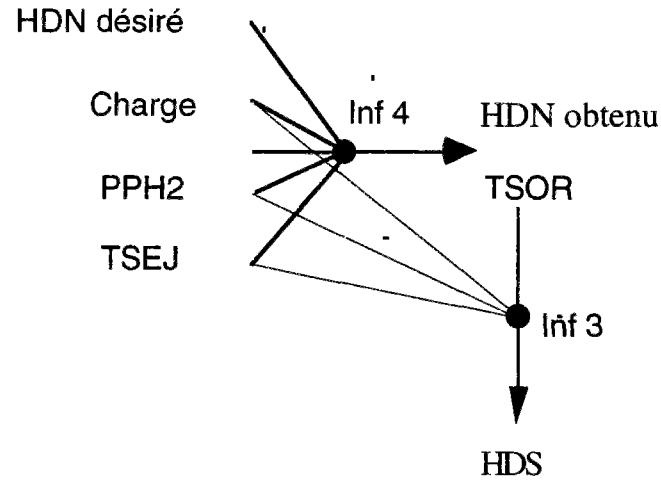
Nous avons simplifié le schéma précédent, en ôtant les flux Charge, PPH2 et TSEJ, qui ne sont pas modifiés.

schéma 26 : **graphe d'enchaînement des inférences simplifié**



Il devient évident qu'une boucle lie TSOR et HDN. Pour résoudre ce point, il faut regrouper ensemble toutes les inférences qui risquent de boucler : les inférences 1 et 2 deviennent l'inférence 4 (notée Inf 4, sur le schéma suivant). Cette inférence 4 va prendre comme entrée les flux HDN, Charge, PPH2 et TSEJ, et produire HDN et TSOR. L'HDN en entrée est renommé HDN désiré et en sortie HDN obtenu, pour pouvoir les distinguer. On retombe alors sur un graphe sans circuit.

schéma 27 : **graphe d'enchaînement sans circuit**



• 3) L'utilisateur a la possibilité d'entrer toutes les données qu'il souhaite. Mais il arrive parfois, que le jeu de données qu'il saisit ainsi soit incohérent.
Par exemple :
Par rapport à la tâche précédente, il entre toutes les informations nécessaires sur la charge, la pression (PPH2), l'HDN et le temps de séjour (TSEJ). Dans ce cas, il y a incohérence : il serait très étonnant que l'application de l'inférence 'calcul de la température de réaction de désazotation' appliquée aux entrées PPH2, Charge, TSEJ et HDN conduisent à la même température que celle que l'utilisateur a entrée. L'application ne sait pas aujourd'hui déduire automatiquement ces cas d'incohérences. La seule méthode possible à l'heure actuelle est de tous les recenser et de les tester systématiquement.

Or, il existe en fait une solution pour contourner cette difficulté et détecter les cas d'incohérence automatiquement. L'idée est ici de créer une liste de tous les chemins possibles, pour partir des données et arriver au résultat. Cette liste dans notre cas n'est pas gigantesque, et peut compter environ un millier d'items. Cette liste doit être structurée : elle doit être indexée par les données d'entrée et les sorties voulues, et elle doit donner le chemin qui permet de passer des unes aux autres.

Une fois cette liste constituée, on procède comme suit pour détecter une incohérence : on compare l'ensemble des entrées disponibles et des sorties désirées avec l'index de la liste. Plusieurs cas se présentent :
• quand on ne trouve pas de ligne dans l'index qui correspond, toutes les données d'entrée ne sont pas fournies.
• quand on trouve une (ou plusieurs) ligne(s) de l'index capable(s) de déduire les sorties à partir d'un sous-ensemble des données, il y a incohérence entre des données surnuméraires.
• quand un seul item de la liste est trouvé, les données sont cohérentes.

Transformer l'ensemble des inférences en cette liste revient à une compilation de la base d'inférences, comme il a été pratiqué dans le cas de EXTRA (cité dans Gondran 93).

Le mécanisme de résolution est alors extrêmement simple : il suffit de chercher dans l'ensemble de ces chemins, celui qui correspond au cas qu'on traite. Puis de le parcourir en exécutant chacune des inférences mentionnées. On est passé d'un mécanisme de chaînage arrière, comme nous l'avons décrit précédemment à un chaînage avant.

CHAPITRE VI

COURBES

VI.1) INTRODUCTION

Nous allons voir, dans ce chapitre, comment l'expert **construit** les courbes, comment il les **valide** et comment l'utilisateur s'en **sert**. Une grande partie de la base de connaissances est en fait constituée de ces courbes. Avant d'être intégrées à PRINCE, celles-ci sont examinées finement par l'intermédiaire des outils mis au point; l'analyse des points expérimentaux est un préalable indispensable à leur exploitation. Une fois acceptée, une courbe est rappelée en affichage durant l'utilisation.

Notre présentation suivra la façon naturelle de procéder des experts, quand ils veulent intégrer de nouvelles connaissances : nous commencerons, comme ces derniers, par analyser les points expérimentaux disponibles pour construire des courbes faisant fonction de corrélations (section 2). Puis nous montrerons comment les experts valident et exploitent les courbes ainsi bâties (section 3).

• Construction des courbes
Dans la seconde section, nous rappellerons d'abord quel est le contexte de notre étude et pourquoi les points expérimentaux sont difficiles à exploiter. Nous étudierons ensuite quels sont les cas difficiles. Il s'agit d'exposer les situations pratiques que les experts veulent examiner, quels sont les outils disponibles pour les aider, et quels sont les différents problèmes auxquels nous nous sommes trouvés confrontés.

Nous verrons alors que pour mettre au point des corrélations, on se base sur des outils d'interpolation et d'optimisation. Quand ces méthodes ne présentent pas de difficulté particulière du point de vue de l'intelligence artificielle, nous avons reporté leur description en annexe, même si les algorithmes ne sont pas immédiats à mettre en place.

Par contre, nous expliquerons plus à fond **deux solutions originales et nouvelles**, pour pratiquer des interpolations. La première utilise les propriétés de monotonie et de concavité des courbes. La seconde s'appuie sur un raisonnement à base de cas. Ces deux façons de faire exploitent les mêmes informations que les experts et semblent donc naturelles à ces derniers.

• Validation et exploitation des courbes
Dans la troisième partie, nous examinerons comment les experts exploitent les connaissances mises à leur disposition. Nous verrons d'abord que les méthodes

mises en place permettent de garantir la **validation** de la base de connaissances, c'est-à-dire d'en assurer la complétude, la cohérence et la pertinence.

Nous détaillerons ensuite quelles sont les différentes méthodes d'affichage et de tracé des courbes. Notamment, nous expliquerons comment utiliser les relations de différentes natures entre les variables (courbes données par des points, formules ou programmes), de façon directe, ou **en les inversant**.

Enfin, nous présenterons rapidement comment l'expert manipule l'interface, dans l'application PRINCE, pour valider les courbes puis les utiliser. On mettra l'accent notamment sur l'**interaction double** qui a été mise en place, permettant à l'utilisateur d'agir sur les courbes à partir de leur définition, ou de leur visualisation à l'écran.

Notre réalisation constitue certainement une base pour la fabrication d'un logiciel d'exploitation des courbes expérimentales. Les fonctionnalités mises en place correspondent précisément à l'attente des utilisateurs. L'accent a été mis en particulier sur une interaction facile entre des manipulations sur les points de base et l'affichage à l'écran des courbes correspondantes. Cette interaction est à double sens, ce qui veut dire qu'on peut agir sur les courbes en modifiant non seulement leur définition, mais aussi en agissant directement sur les graphiques, par l'intermédiaire de la souris .

Quelques travaux antérieurs

Bertrand Rousseau (Rousseau 1988) détaille une application informatique, qui présente des **similarités** avec celle que nous détaillons ici. D'une part, il expose une architecture en tâches qui s'enchaînent pour produire un raisonnement. Ces tâches sont plutôt d'ordre mathématique (comme, par exemple, *la recherche des points fixes d'un système d'équations différentielles*). D'autre part, il décrit un **environnement graphique** destiné à trouver les modèles qui s'adaptent le mieux à un ensemble de points expérimentaux de base, dans le domaine de la biométrie, son domaine d'application. A ce titre, il dispose d'équations prédéfinies qu'il faut ajuster au mieux, par optimisation. Le choix des modèles de base se fait graphiquement, c'est-à-dire qu'on visualise immédiatement les courbes obtenues, et c'est l'utilisateur qui décide du meilleur modèle.

Une application développée chez Thomson (Plasson & 91) développe également cet environnement de modules (ou tâches) qui doivent s'enchaîner, et de visualisation graphique, pour examiner les résultats. L'une des grandes originalités de cette application est d'organiser la liaison entre les modules sous forme graphique : l'utilisateur crée à l'écran les liens (sous forme de traits) entre les différents traitements (représentés par des boîtes). Puis il exploite à l'écran les courbes obtenues pour choisir les meilleures simulations.

En fait, dans le domaine scientifique et de la recherche, le groupement, dans une même application, des aspects raisonnement et visualisation des résultats, correspond, de façon intime, à l'activité des ingénieurs. C'est bien évidemment le choix qui est fait dans PRINCE.

VI.2) ANALYSE DES POINTS EXPÉRIMENTAUX

VI.2.1) DES POINTS DIFFICILES A EXPLOITER

VI.2.1.1) UNE GRANDE VARIÉTÉ DE MESURES

Les expériences pilotes produisent des données nombreuses, extrêmement variées, mais malheureusement assez peu comparables. On fait en effet énormément de mesures autour d'un essai pilote. Mais entre deux mesures, souvent plusieurs paramètres ont bougé. Les méthodes de mesure elles-mêmes évoluent, ainsi que les appareillages expérimentaux.

Par ailleurs, les expériences elles-mêmes sont rarement reproduites. Pour se placer dans des conditions stables de fonctionnement sur un pilote, quelques heures (parfois une journée) sont nécessaires. Si plusieurs niveaux de conditions opératoires sont demandés, la durée du programme expérimental complet peut atteindre plusieurs semaines, durée qu'il n'est jamais facile de dégager, sans parler du coût d'une telle campagne de mesures.

Les essais pilotes eux-mêmes sont délicats à mettre en oeuvre, ils dépendent de très nombreux facteurs. Il peut arriver qu'un essai soit raté, sans qu'on se rende compte immédiatement du caractère anormal de la campagne de mesures.

Enfin, les données elles-mêmes sont entachées d'erreur expérimentale et d'incertitudes de mesure.

Ceci dit, ce tableau pessimiste peut être nuancé : le soin apporté à ces essais est très grand et l'expérience acquise au cours de 25 ans de fonctionnement permet de combler beaucoup de lacunes expérimentales. Les mesures vont suivre certaines règles d'évolution connues. Il serait par exemple étonnant, qu'après avoir observé pendant des années qu'un accroissement de pression permettait de diminuer la teneur en soufre mesurée dans l'effluent d'un procédé d'hydrodésulfuration, on remarque tout d'un coup une tendance contraire.

VI.2.1.2) PEU DE MODÈLES

Il faut noter également que dans le domaine des procédés pétroliers et particulièrement du raffinage, peu de modèles sont disponibles.

• 1ère raison) Comme nous l'avons déjà souligné, les charges pétrolières peuvent contenir plusieurs dizaines de milliers de types de molécules, qui ont, toutes en principe, un comportement différent. Et même si la charge de départ contient un nombre limité de types de molécules différents, comme dans le cas du reforming, ou dans certains procédés de pétrochimie, il faut encore compter avec les isomères, c'est-à-dire des molécules qui ont la même formule mais dont les atomes sont placés différemment, et qui ont donc des comportements chimiques originaux.

• 2ième raison) Les données disponibles pour établir les modèles ne sont pas toujours fiables. Les qualités mesurées sont parfois le résultat de méthodes presque artisanales. Il faut se souvenir que ces qualités doivent pouvoir être obtenues dans n'importe quel endroit du globe, quel que soit l'environnement scientifique du pays. Par exemple, le point de fumée d'un kérosène est la hauteur de la flamme, jusqu'au point où apparaît la fumée, obtenue dans un appareil qui ressemble (à s'y méprendre) à une lampe à kérosène qu'on utilisait autrefois. Cette hauteur est mesurée en mm à l'oeil nu.

• 3ième raison) Les conditions qui régissent un procédé catalytique sont mal connues. Il y intervient des phénomènes complexes, tel que l'écoulement d'un fluide au sein d'un catalyseur, qu'on commence tout juste à étudier. La thermodynamique, quand il s'agit de mélanges complexes, est peu maîtrisée. Et pour encore compliquer le tout, la cinétique des réactions n'est pas toujours complètement clarifiée, même dans les cas les plus simples : une réaction peut se dérouler en une nanoseconde, on imagine les difficultés de mesure.

VI.2.1.3) DES MODÈLES NON LINÉAIRES

Les modèles sont souvent non linéaires. A titre d'illustration, considérons la loi d'Arrhenius, qui modélise les effets de la température sur les réactions chimiques. Cette loi, au départ empirique (voir l'annexe de ce chapitre), a la forme suivante :

$$k = k_0 * e^{(\frac{-E}{R \cdot T})}$$

où k est appelée constante cinétique,
k_0 est le facteur de fréquence,
E est l'énergie d'activation,
R est la constante des gaz parfaits,
T est la température absolue en degrés Kelvin.

Il est important de remarquer que dans le cas des procédés pétroliers, E peut être de l'ordre de 30000 calories par mole, T de l'ordre de 600 degrés Kelvin, tandis que, dans ce système d'unité, R est voisin de 2 calories par mole et par degré K. En conséquence, si k_0 est de l'ordre de l'unité, k vaudra environ :

$$1 * e^{+(\frac{30000}{2*600})} \cong e^{25}$$

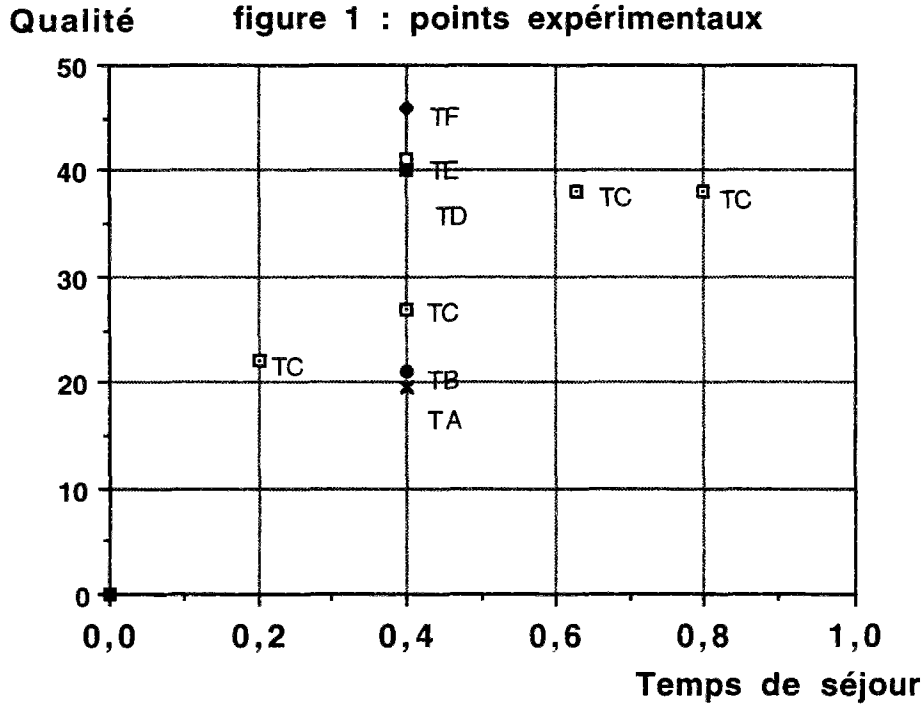
La moindre variation de la valeur de l'exposant de l'exponentielle (ici 25) a de fortes répercutions sur la concentration finale. Cette loi conduit en combinant les diverses réactions qui ont lieu dans le réacteur soit en parallèle, soit en séquence, à des systèmes d'équations, qualifiées de 'raides', difficiles à manipuler et à ajuster.

VI.2.2) ÉTUDE DES CAS

VI.2.2.1) TRACER DES COURBES AVEC PEU DE POINTS

VI.2.2.1.1) tracer une famille de courbes

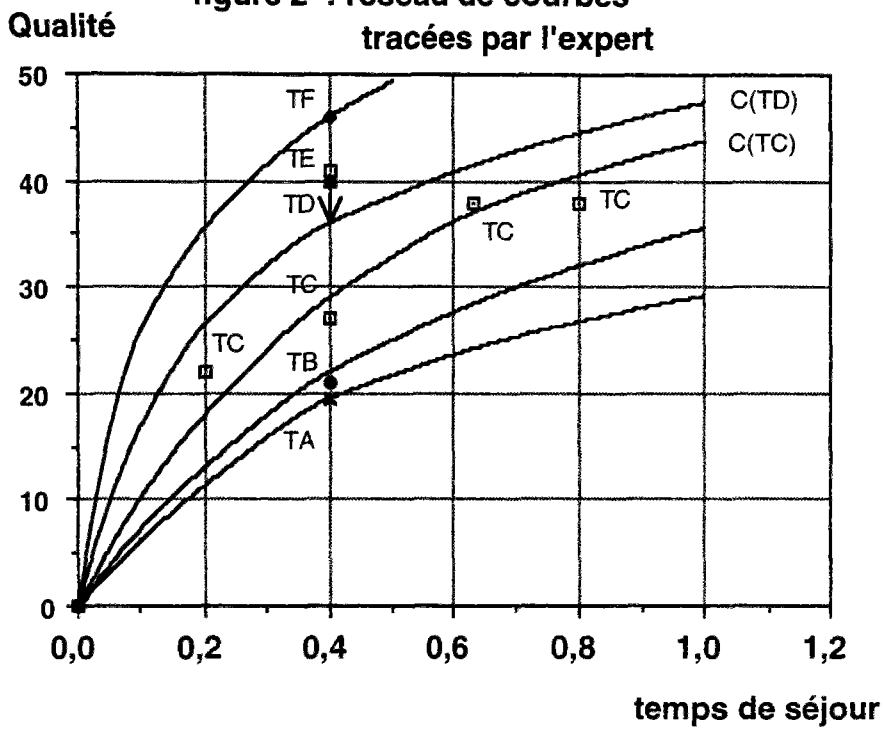
Les trois figures suivantes illustrent un problème qui s'est réellement posé. Pour une charge donnée, après transformation dans un procédé P, une certaine qualité de l'effluent (appelons-la Q) a été mesurée, à différentes VVH, et différentes températures (TA, TB, TC, TD, TE, TF). L'ensemble des points disponibles a été retranscrit sur un graphe qui a pour abscisse, le temps de séjour (c'est-à-dire l'inverse de la VVH), et pour ordonnée, la qualité Q (figure 1).



L'ingénieur spécialiste de ce procédé a alors tracé les courbes de la figure 2. Examinons comment il est passé de la figure 1 à la figure 2.

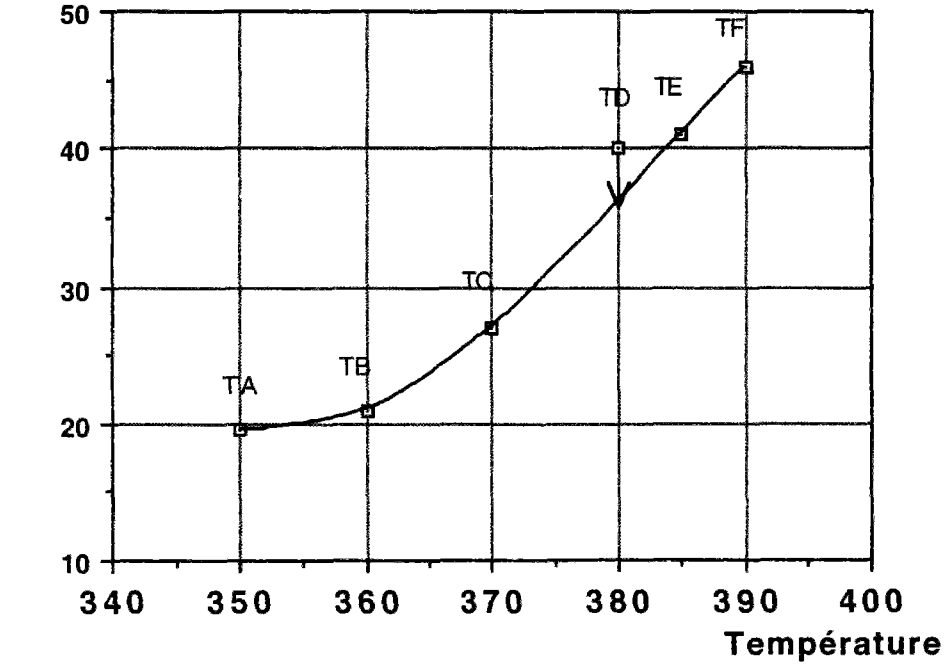
Par définition de la qualité Q, toutes les courbes passent par l'origine. Par expérience, notre ingénieur sait que les courbes sont a priori monotones croissantes, et à concavité négative. Prenant alors son pistolet (règle recourbée, dessinée pour aider à tracer des courbes), l'ingénieur a sélectionné la courbe pour laquelle il avait le plus de points expérimentaux et a dessiné la courbe marquée TC, qui exploite les points à la température TC. Toutes les autre courbes en ont été déduites par analogie.

figure 2 : réseau de courbes tracées par l'expert



Pour vérifier ce premier essai, une projection sur le plan (Température, Qualité) a été entreprise à VVH = 0,4 heure^{-1} (figure 3).

figure 3 : courbe projetée à VVH = 0,4 h-1



Pourquoi cette VVH ? Parce que c'est à cette valeur que nous disposons du plus de points. Sachant que la courbe obtenue doit être monotone croissante (voir

figure 3), et sans point d'inflexion, le point TD marqué d'une flèche sur les figures 2 et 3 a été corrigé, pour obtenir finalement la courbe C(TD) de la figure 2.

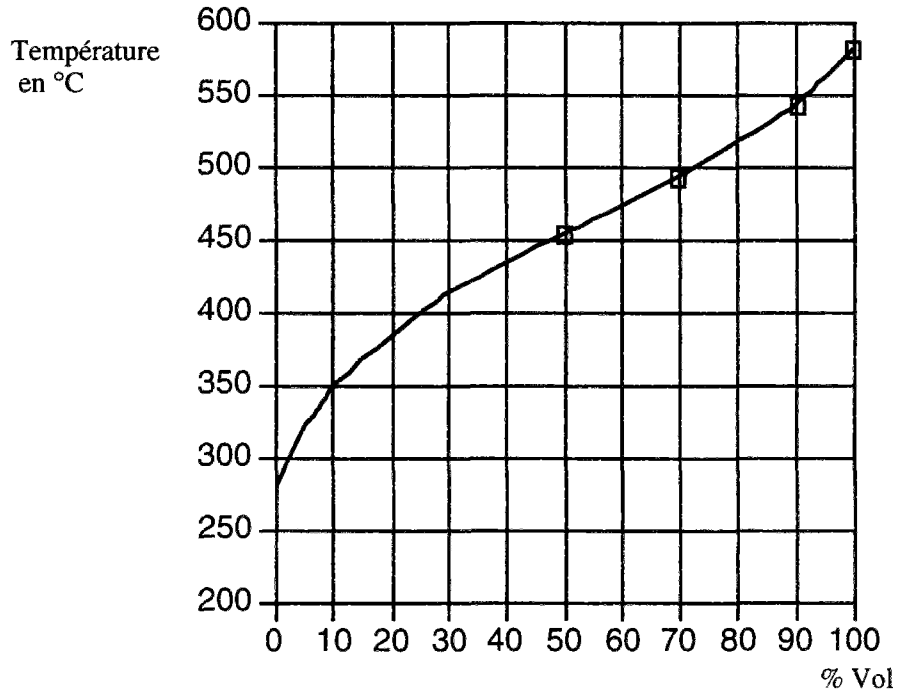
VI.2.2.1.2) tracer une courbe dont on connaît l'allure

Les courbes de distillation présentent l'avantage d'avoir une allure connue.

Bien souvent, les ingénieurs qui s'occupent des procédés de raffinage pétrolier, doivent reconstituer des courbes de distillation, à partir de quelques points. Par exemple, dans l'appel d'offre que nous avons présenté au chapitre III *étude de cas*, tous les points de la courbe de distillation ne figurent pas. Rappelons qu'une courbe de distillation permet de suivre l'évolution du pourcentage (poids ou volume) de la fraction distillée par rapport à la coupe de départ en fonction de la température (voir chapitre II, *contexte et problèmes*, section 1.5).

A partir des points dont nous disposons ici ([50%, 454°C], [70%, 493°C], [90%, 543°C], [100%, 582°C]), voici la courbe qu'un ingénieur spécialiste de ces problèmes de raffinage, trace :

figure 4 : **courbe de distillation tracée 'à la main', à partir des 4 points précédents**



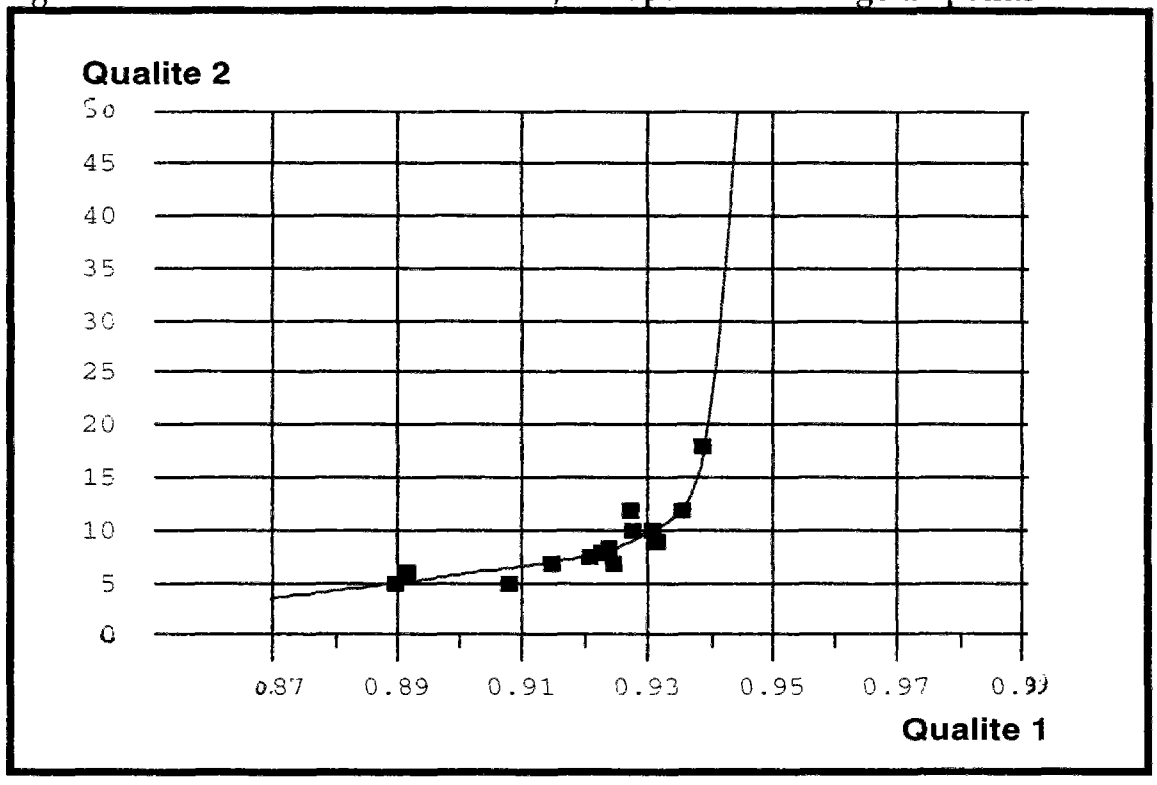
Comment a-t-il fait ? De façon évidente, cet ingénieur s'appuie sur son expérience de dizaines de cas similaires pour se donner une idée des points manquants, ce qui lui permet d'obtenir les informations qu'il désire. Il ne s'agit pas ici de déduire la courbe exacte, des incertitudes sont admises et intégrées dans le raisonnement de l'ingénieur.

Nous avons développé une méthode spécifique, que nous exposerons dans la partie VI.2.3, *quelques solutions*, de ce chapitre. Cette méthode, adaptée au cas des courbes de distillation, peut se généraliser à d'autres situations semblables. Elle consiste à tirer parti des cas similaires, pour former une nouvelle courbe, un peu comme dans les raisonnements à base de cas.

VI.2.2.2) TRACER DES COURBES AVEC UN NOMBRE SUFFISANT DE POINTS, MAIS PAS DE MODÈLE

La figure 5 illustre un autre cas : les données recueillies sont nombreuses (ou suffisamment nombreuses), mais aucun modèle n'est disponible.

figure 5 : **courbe tracée dans PRINCE, s'adaptant à un nuage de points**



Dans cet exemple, deux qualités de charge sont a priori corrélées. La sortie graphique présentée dans cette figure est obtenue à partir de l'application PRINCE. A la main, on aurait du placer le pistolet et le déplacer au mieux, pour que la courbe obtenue ait une allure correcte.

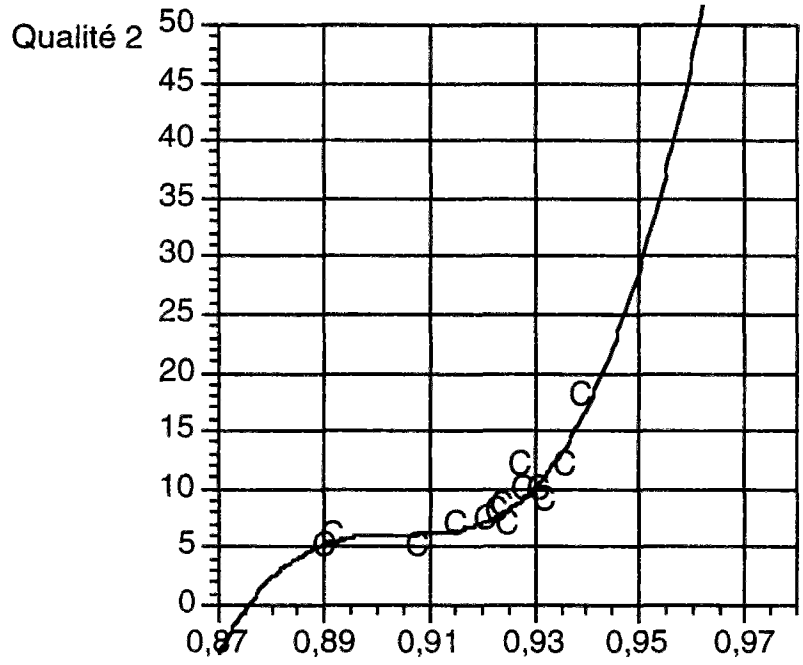
Il est aisé, face à ce problème, d'ajuster une courbe d'équation polynomiale. Ce genre de facilités est à la disposition des ingénieurs dans le cadre des logiciels commerciaux classiques. Mais les résultats sont parfois décevants (figure 5-bis).

En effet dans les valeurs basses, aucun point d'inflexion n'est observé dans la réalité. Même une légère extrapolation de ce modèle s'avère dangereuse. D'autres méthodes peuvent être mises en oeuvre (notamment une optimisation sous contraintes), que nous évoquerons dans la partie QUELQUES SOLUTIONS.

figure 5-bis : **courbe d'équation**

$f(x) = 239813{,}4\,x^3 - 650787{,}4\,x^2 + 588693{,}3\,x - 177504{,}1$

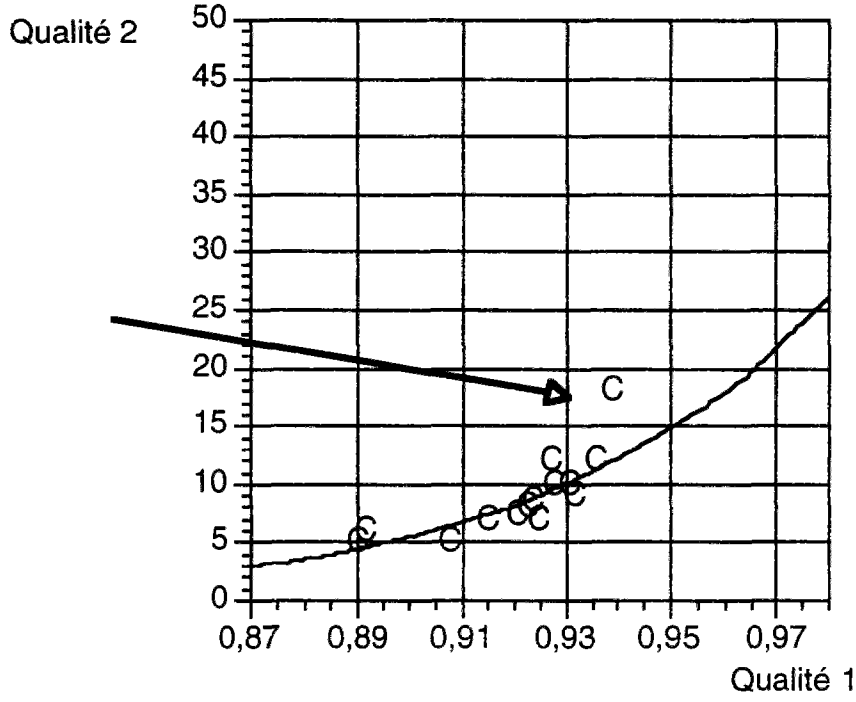
La précision des nombres est celle que donne le logiciel utilisé.



Nous avons essayé à titre d'exemple un modèle en puissance. (voir figure 5-ter). Que conclure ? Dans ce dernier cas, en examinant le tracé, un point (marqué d'une flèche) peut apparaître aberrant. Mais en fait, le tracé de la figure 5 corrobore mieux la connaissance expérimentale acquise.

Le plus souvent, l'ingénieur qui fait ce travail a quelques informations supplémentaires sur la courbe à obtenir, que l'application brutale de formules mathématiques choisies a priori a tendance à ignorer. Même sans disposer de formules ou de modèles analytiques, le spécialiste connaît souvent le sens de monotonie du tracé (courbe croissante ou décroissante), son sens de concavité, son comportement à l'infini (la courbe a tendance à s'écraser), ...

figure 5-ter : **courbe d'équation**
$f(x) = 37{,}85218\ x^{18{,}21948}$



VI.2.2.3) UN MODÈLE EST DISPONIBLE ou POSE A PRIORI

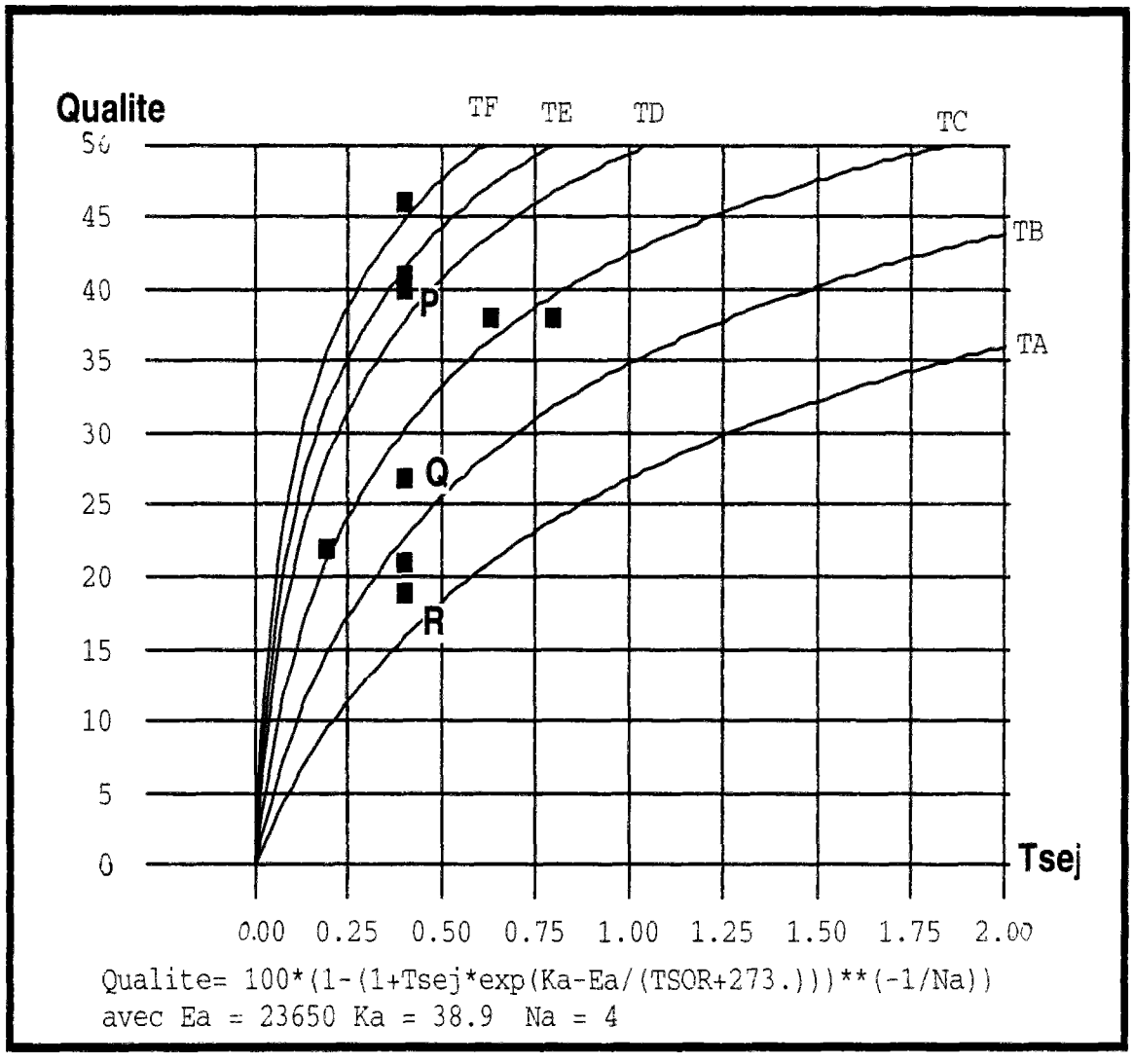
Lorsqu'un modèle est disponible, les techniques mathématiques classiques sont utilisables : régression linéaire, ou optimisation quand l'expression à ajuster n'est pas linéaire.

Dans toute la suite, on sait que l'on fait des approximations et que l'on a des incertitudes que l'expert connaît.

Dans certains cas, le modèle obtenu peut être amélioré en écartant les points manifestement aberrants, comme dans l'exemple de la figure 3.

Il peut arriver aussi qu'un modèle posé a priori conduise à des **conclusions difficilement interprétables**. L'exemple de la figure 6 repart des mêmes points, exactement, que ceux de la figure 3.

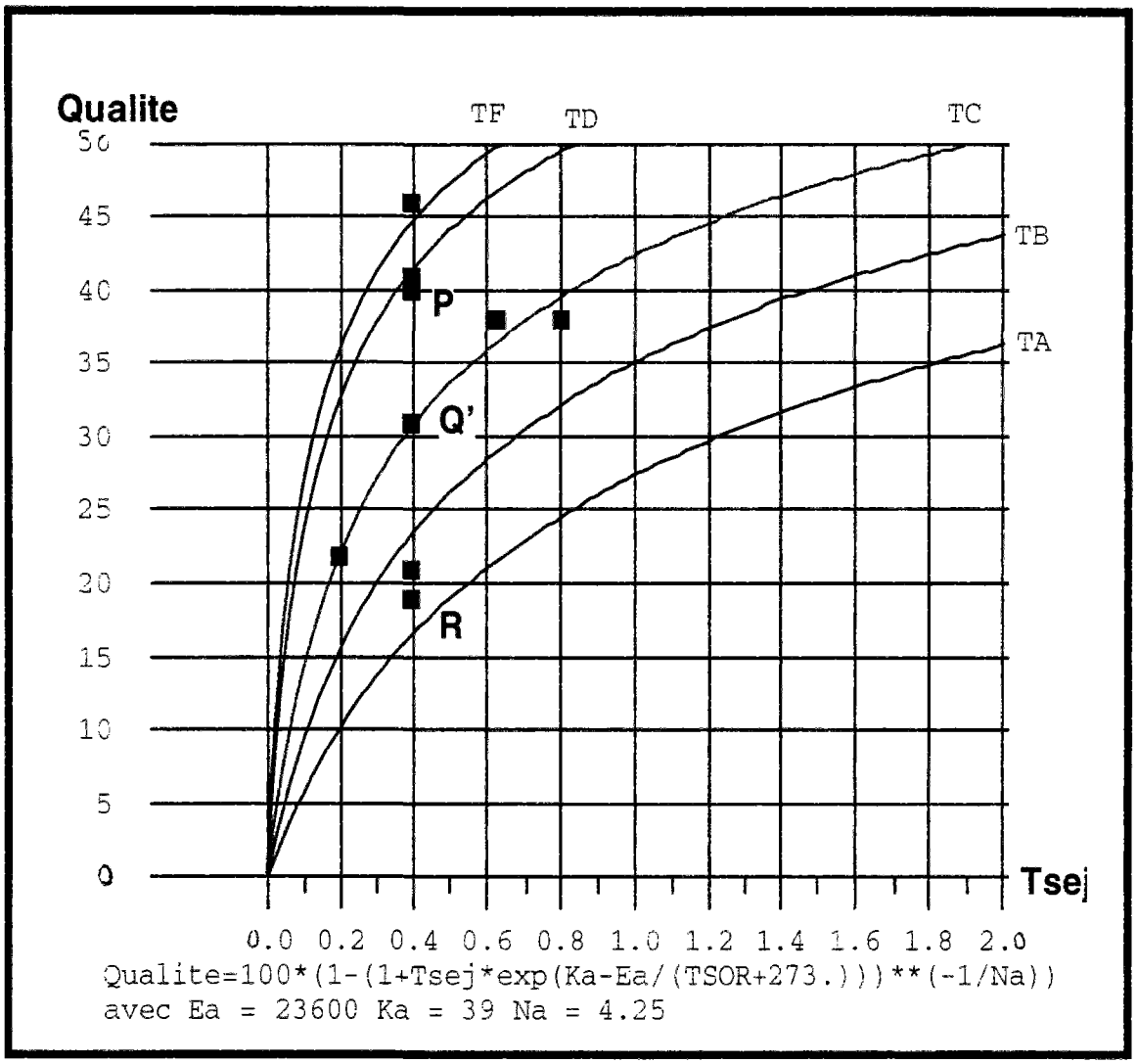
figure 6 : **premier essai d'optimisation.**
Tous les points sont supposés valides



L'optimisation, menée dans l'application PRINCE, a conduit à l'équation ci-dessus. Cette équation s'inspire de l'équation de la cinétique, une fois intégrée, et la constante k a été remplacée par l'expression issue de la loi d'Arrhenius, et mise sous une forme exponentielle.

A priori, les points supposés aberrants sont les points les plus éloignés des courbes. En considérant l'abscisse TSEJ = 0,4, les points P, Q, R sont douteux. L'examen de la figure 6 conduit à supposer que le point Q est douteux (comme étant le point dont la distance à la courbe correspondante -TC- est la plus grande). Or, en fait, c'est le point marqué P qui a été remis en question par les experts, par l'intermédiaire de la figure 3. De plus, si l'on corrige le point Q, pour le ramener au point Q' (voir figure 6-bis, page suivante), et que l'on fait tourner l'optimisation en éliminant le point P comme les experts, on obtient une réponse très voisine de la précédente, à partir de laquelle il est encore une fois délicat de tirer des conclusions.

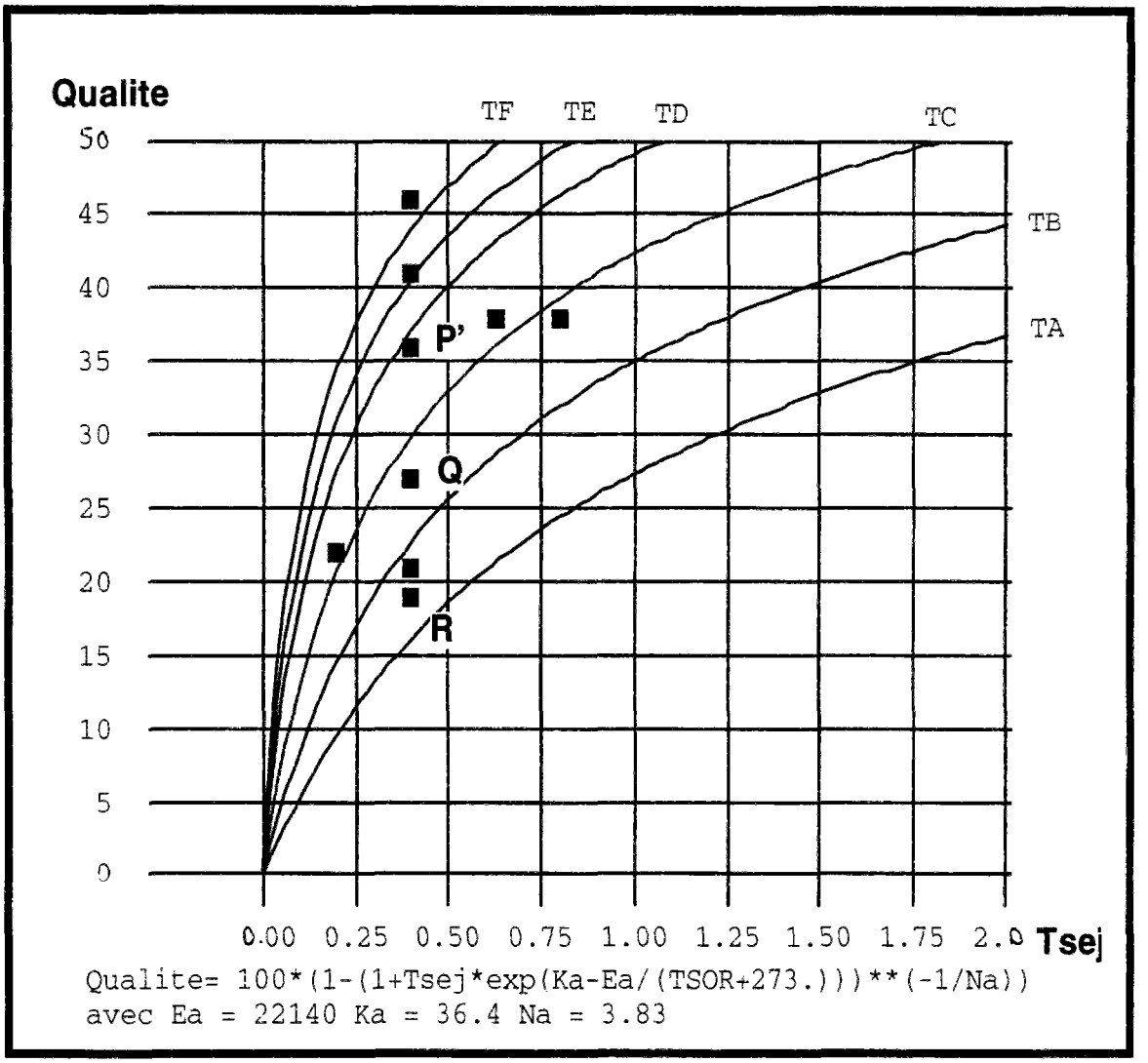
figure 6-bis : **second essai d'optimisation**
Le point Q est ramené au point Q'.



Que conclure de cet exemple ? Premièrement, il est fort possible que le modèle supposé soit inadéquat et ne convienne pas à ce cas. Malheureusement, c'est le seul modèle actuel disponible. Secondement, toutes les informations des experts n'ont pas été exploitées. En particulier, la courbe de la figure 3, qui reprend l'ensemble des points de la figure 6, à VVH = 0,4 heure^{-1}, doit être croissante (la qualité augmente quand la température augmente), et à concavité positive (la qualité augmente d'autant plus que la température croit). Ces deux constations issues de l'expérience sont suffisantes pour écarter le point P.

Nous avons refait un essai en respectant l'avis des experts, c'est-à-dire en corrigeant le point P en P' (figure 6-ter). Dans ce cas, le point R semble de nouveau controversé, et ceci contrairement à l'avis des experts (mais ce point est peut-être erroné, sans que les experts l'aient détecté).

figure 6-ter : **troisième essai d'optimisation**
Le point P est ramené au point P'



Finalement, nous pouvons conclure que seuls les experts sont capables de poser un modèle a priori. Ceci dit, dès que plusieurs dimensions interviennent, il est très difficile, à la main, de rendre toutes les courbes dans toutes les dimensions cohérentes. Seul un modèle posé a priori est capable à peu de frais de parvenir à ce résultat.

C'est pourquoi **il est nécessaire d'exploiter toutes les informations disponibles**, sans qu'aucune ne soit exclusive :
• des projections dans les dimensions où le plus grand nombre de points est disponible, des considérations sur le sens de monotonie et la concavité des courbes pour écarter les points aberrants,
• l'exploitation d'un modèle, inspiré de l'expérience pour tracer le réseau de courbes et éviter les incohérences.

VI.2.2.4) RÉSUMÉ DES PROBLÈMES

Raisonner sur des points demande a priori un outillage informatique et mathématique conséquent.

• D'abord, il faut savoir faire passer une courbe d'allure visuelle correcte par un ensemble de points de définition donnés. Il s'agit d'interpoler les points correctement. Ces courbes sont en particulier affichées à l'écran, après calcul, et tout défaut est immédiatement apparent.

Par conséquent, les courbes ne peuvent pas être linéaires par morceaux, cela se voit à l'affichage. Si l'on souhaite utiliser sur chaque segment de courbe des paraboles, on se heurte à des problèmes de maximum, ou de minimum. On n'est jamais trop sûr qu'un extremum de la parabole ne va pas se situer en plein milieu du segment à tracer. Les courbes de Bézier (Bézier 87) supposent quant à elles que l'on connaisse a priori les tangentes en chaque point. Or ce n'est pas le cas, seuls les points expérimentaux sont connus et non leurs tangentes. D'autant plus que l'on ne maîtrise pas directement la concavité des courbes de Bézier.

Nous avons utilisé pour obtenir des tracés corrects, l'**interpolation de Stineman**, qui est adaptée à nos cas d'étude, que nous avons légèrement retravaillée et que nous présentons plus longuement en annexe de ce chapitre.

• Pour tracer une courbe dans un nuage de points (il ne s'agit plus ici d'interpolation, mais de lissage), des ajustements simples d'équations polynomiales ou d'exponentielles ne suffisent pas.

Nous avons développé un algorithme nouveau, capable de **dessiner une courbe d'allure donnée, sans disposer de modèle**. Par allure, nous entendons concavité et/ou monotonie. Cet algorithme est expliqué dans le paragraphe QUELQUES SOLUTIONS.

• Quand on dispose d'un **modèle de courbes, mais pas d'équation**, comme dans le cas des courbes de distillation, nous avons développé une méthode adaptée, que nous exposerons dans la partie QUELQUES SOLUTIONS, tracer des courbes à partir de modèles.

Cette méthode suit, en la simplifiant, la démarche du raisonnement à base de cas. Plutôt que de s'appuyer sur une analyse complètement formalisée des courbes de distillation, on part d'une base d'exemples, dont on tire certains exemples pertinents, qu'on adapte au cas traité.

• Il faut ensuite savoir ajuster une courbe d'équation imposée sur un ensemble de points. Ceci est nécessaire non seulement pour utiliser les modèles disponibles, mais aussi pour tracer des courbes d'allure donnée dans un ensemble de points, puisqu'à la base ces courbes auront des équations génériques qu'il suffira de particulariser sur nos exemples. Ainsi, une exponentielle est une courbe monotone croissante.

Si l'équation de base est linéaire, les méthodes numériques de régression linéaire multivariables sont celles que nous allons employer. Ces méthodes bien connues sont très faciles d'emploi.

Quand l'équation est non linéaire, nous avons choisi d'adapter la méthode d'optimisation de Levenberg-Marquardt, recommandé par le recueil de méthodes numériques 'Numerical Recipes in C' (Press & 88). Nous ne développerons pas cette méthode dans ce mémoire, parce qu'elle figure classiquement dans ce type de manuel, il suffit de se reporter à cette référence pour avoir plus de précision.

Les équations que l'on va chercher à employer sont variables par essence, puisqu'elles dépendent du problème, que l'on ne connaît pas d'avance. C'est pourquoi elles doivent être saisies sous leur forme habituelle (forme dite préfixée, voir annexe) et transformée sous un format compréhensible par les programmes informatiques. Nous nous sommes servis d'algorithmes bien connus pour transformer une formule saisie sous forme infixée en arbre syntaxique, qui est la forme d'évaluation habituelle des expressions numériques, ce qui permet de faire tourner les routines d'**optimisation** choisies à partir de n'importe quelle équation symbolique fournie par l'utilisateur.

• Écarter les **points aberrants** reste toujours une préoccupation majeure face à des résultats expérimentaux. Pour cela nous n'avons pourtant pas développé de module de raisonnement, mais plutôt mis en place les outils nécessaires pour que les experts puissent intervenir eux-mêmes facilement.

Comme nous avons pu le remarquer, les informations nécessaires à cette tâche, en l'absence de modèle, sont extrêmement variables. L'exemple de la figure 5 et ceux des figures 6 sont significatifs à cet égard : le fait de supposer un modèle a priori change considérablement les conclusions que l'on peut tirer. Par exemple, quand on fait l'hypothèse d'une formule en puissance un point aberrant apparaît sur la figure 3.

Pour mener cette fonction à bien, il faut chercher à exploiter toutes les informations disponibles : un modèle quand on en dispose, et en l'absence de modèle, essentiellement les sens de variation d'un paramètre par rapport à l'autre. Par exemple, *quand la température monte, l'HDN observé est plus élevé, et suit une courbe monotone croissante de concavité positive.*

VI.2.3) QUELQUES SOLUTIONS

VI.2.3.1) LISSAGE SELON DES CRITÈRES DE MONOTONIE ET DE CONCAVITÉ

VI.2.3.1.1) Exposé du problème

Les exemples des figures 5, 5-bis et 5-ter illustrent le problème auquel un ingénieur qui cherche à exploiter des points expérimentaux est bien souvent confronté. La figure 5 montre un exemple de résultats obtenus, en appliquant la méthode que nous allons exposer.

Ayant reporté sur un diagramme d'axe X et Y les points expérimentaux, un simple coup d'oeil suffit au spécialiste, en l'absence de modèle analytique, quand il n'a pas d'a priori, pour se faire une idée du type de courbe qu'il souhaite dessiner. Le plus souvent, il s'agit de **courbes simples qui ont une monotonie et/ou une concavité données.**

En pratique, huit cas se présentent. Les quatre premiers cas sont les plus fréquents. Les courbes peuvent être :
- monotone croissante et à concavité positive (tournée vers le haut).
- monotone croissante et à concavité négative (tournée vers le bas).
- monotone décroissante et à concavité positive.
- monotone décroissante et à concavité négative.
- à concavité positive
- à concavité négative.
- monotone croissante
- monotone décroissante.

Les formules mathématiques s'ajustent mal à ce type de problèmes. Les figure 5 et 5-bis illustrent deux essais qui donnent des résultats contestables. D'autant plus, que ces formules n'ont pas de sens physique, et ne sont pas forcément reproductibles d'un cas à un autre. Par exemple, dans la figure 5-ter, un modèle à base d'exponentielle, de la forme : $y= \alpha (e^{\beta x}-1)$, a été essayé. Ces modèles sont sympathiques au départ, parce que leur concavité et leur sens de variation sont parfaitement maîtrisables. Cependant, ils sont parfois difficiles à faire converger et ne sont pas toujours satisfaisants : l'examen de la figure 5-ter suggère que le point marqué d'une flêche est aberrant, alors que ce n'est pas le cas.

Optimiser des équations polynomiales sous contraintes n'est pas non plus suffisant :
- d'abord, les contraintes ne sont pas nécessairement faciles à exprimer, et ne s'écrivent pas toujours sous la forme de relations linéaires entre les paramètres à déterminer.
- puis, imposer une forme à une courbe d'équation polynomiale, même contrainte, n'est pas forcément immédiat; comment dire par exemple que la tangente à l'origine ne doit pas être trop plate ?
- comment faire pour que le logiciel d'optimisation déduise la forme que l'on cherche à imposer de lui-même, par simple examen des données.

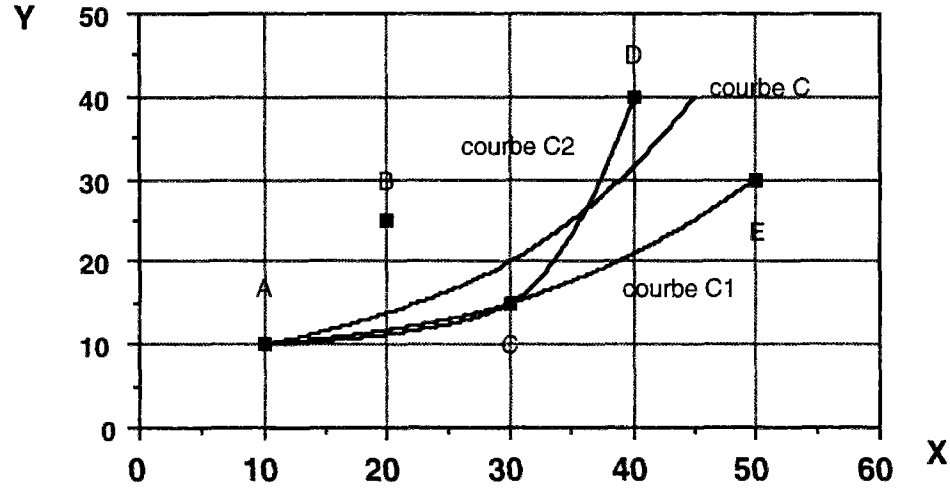
- ensuite, pour que la procédure soit automatique, il faut savoir initialiser l'optimisation dans toutes les conditions. Et il est difficile de prévoir toutes les situations envisageables, qui dépendent du degré du polynôme utilisé, de l'allure souhaitée de la courbe, de la forme du nuage des points initiaux, de la présence ou non de points aberrants, ...
- enfin, ces équations doivent savoir s'adapter à toutes les allures de courbes, et ce n'est pas évident a priori. Par exemple, la courbe désirée peut être très plate, puis 'tourner' brusquement et croître ensuite gentiment, ce qui est plus facile à dire qu'à transcrire en termes d'équation polynomiale.

VI.2.3.1.2) Solution

Pour répondre à ce problème, nous avons mis au point la méthode suivante, qui mélange des **aspects qualitatifs et quantitatifs**. L'idée de départ est de chercher parmi les points présents, la (ou les) ligne(s) qui respecte(nt) les **contraintes de monotonie et de concavité** que l'on s'est donné, et qui passe(nt) par **le plus grand nombre de points** possibles (critère qualitatif), puis de l'ajuster numériquement (critère quantitatif).

Ces lignes ne sont pas données par une (ou des) formule(s) mathématique(s), mais au contraire, résultent d'une interpolation entre les points de base. Voici un exemple simple :

figure 7 : **un exemple arbitraire simple**



Critère qualitatif (génération des courbes candidates) :

parmi les points A, B, C, D et E, on cherche des **lignes monotones croissantes, et à concavité positive**, et passant par le maximum de points, . Il en existe deux, la courbe C1 passant par les trois points A, C, E, et la courbe C2, passant par les points A, C, D.

Critère quantitatif (ajustement numérique) :

chaque ligne de points définit une fonction y = f(x), par **interpolation** entre les points et **extrapolation** aux bornes. Cette procédure d'interpolation/extrapolation ne fait pas nécessairement appel à des

fonctions linéaires. Nous avons utilisé pour notre part l'algorithme de Stineman.

Pour chaque fonction f et donc pour chaque ligne de points générée, nous calculons les paramètres λ et μ, qui permettent de minimiser le critère suivant, calculé sur l'ensemble de tous les points du nuage :

$$\sum_{i=1}^{n} [Y_i - (\lambda\ f(X_i) + \mu)]^2$$

où X_i et Y_i représentent respectivement l'abscisse et l'ordonnée du point i, n est le nombre de points du nuage,
λ et μ sont des constantes réelles à déterminer.

En fait, il s'agit de légèrement déformer la ligne considérée pour qu'elle s'ajuste mieux sur l'ensemble de tous les points du nuage. Nous obtenons ainsi autant de nouvelles courbes d'équation $y = \lambda f(x) + \mu$, que de lignes initiales à faire bouger, d'équation $y = f(x)$.

On retiendra finalement la courbe qui passe le plus près du plus grand nombre de points. Il est immédiat de se rendre compte qu'il s'agit de la courbe qui rend le critère précédent, le plus faible.

Si, au lieu de chercher des courbes dont les sens de monotonie et de concavité sont donnés, on avait envisagé le cas d'une courbe dont seule la concavité est imposée, l'algorithme précédent se complique légèrement sur le plan conceptuel : on cherche à se ramener au cas précédent. Sur le plan pratique, la mise en oeuvre est délicate et sera exposée à part.

VI.2.3.1.3) Algorithme

L'algorithme qui sous-tend cette méthode comprend trois étapes.

• 1) Génération des lignes ou comment chercher les suites de points répondant aux critères de monotonie et de concavité donnés.

Une suite de points permet, en appliquant la méthode d'interpolation retenue, d'obtenir une fonction $y = f(x)$, définie sur tout le domaine, qui passe par les points de la suite, et qui définit une ligne.

Savoir reconnaître une suite de points monotones est immédiat. La détermination de la concavité s'appuie, quant à elle, sur une comparaison des pentes successives entre les points : une courbe à concavité positive (respectivement négative) est une courbe dont les pentes croissent (resp. décroissent).

L'algorithme qui permet de chercher une ligne de ce type dans un nuage de points est raisonnablement consommateur en temps de calcul : on démontre au

paragraphe VI.2.3.1.4 que le temps de calcul est de l'ordre du nombre de points du nuage au cube.

Le nombre de lignes généré par l'algorithme précédent peut être exponentiel dans les cas les plus défavorables (plusieurs lignes sont générées à chaque itération de l'algorithme) mais ces cas sont impossibles à obtenir dans des conditions normales d'utilisation (voir VI.2.3.1.5).

• 2) Élimination ou comment retenir les lignes de base.

On isole les lignes qui passent par un grand nombre de points .et qui comprennent au moins trois points non alignés (sinon, on recherche une droite, et une régression linéaire simple suffit). Les autres lignes sont éliminées. Concrètement, si N est le nombre de points de base d'une ligne, on recherche $N_{max}=\max_{lignes}(N)$, et on garde les lignes pour lesquelles $N \geq N_{max}-1$.

Choisir parmi les lignes est indispensable. L'idée est de garder les courbes dont le tracé respecte les critères que l'on s'est fixé, et non de trouver 'la' meilleure courbe. Dans le cas contraire, le critère de sélection, appliqué en troisième étape, risque de ne plus être efficace, et de conduire à des choix inadéquats.

• 3) Test ou comment ajuster une ligne

Actuellement, les lignes retenues passent par certains des points du nuage. Nous allons les ajuster, c'est-à-dire les déformer pour qu'elles tiennent compte de l'influence de tous les points. Pour cela, pour chaque ligne retenue à l'étape 2, on évalue les paramètres λ et μ du critère que nous avons mentionné précédemment (voir paragraphe VI.2.3.1.2), et que nous rappelons ci-dessous :

$$\sum_{i=1}^{n} \left[Y_i - (\lambda f(X_i) + \mu) \right]^2$$

λ et μ sont déterminés par la méthode des moindres carrés. La courbe finalement retenue est celle qui rend ce critère minimum. Plus précisément,
si $([X_1,Y_1], [X_2,Y_2], ..., [X_n,Y_n])$ est une suite de points définissant une courbe,
on considère la nouvelle courbe définie par les points $([X_1,Y'_1], [X_2,Y'_2], ..., [X_n,Y'_n])$,
où $Y'_i = \lambda Y_i + \mu$, pour tout i (λ et μ prenant une valeur différente pour chaque courbe).

A l'expérience, on constate que la méthode proposée convient à des **courbes monotones, croissantes ou décroissantes**. qui sont celles que rencontrent classiquement des ingénieurs, qui font des essais expérimentaux.

Ce test donne de moins bons résultats quand il s'agit de courbes dont seule la concavité est donnée. Intuitivement, le critère d'évaluation choisi ne permet plus d'ajuster indépendamment les branches croissantes ou décroissantes des

courbes, ce qui fait que les changements obtenus sur les courbes sont minimes. Ce cas est traité au paragraphe VI.2.3.1.5.

VI.2.3.1.4) Détail de la génération des lignes

Proposition : le nombre d'alternatives explorées par l'algorithme de génération des lignes est de l'ordre du cube du nombre de points présents dans le nuage. Nous allons traiter le cas des courbes croissantes et convexes, la généralisation aux autres cas est immédiate.

Hypothèse : Considérons un nuage de n points $M_i(x_i, y_i)$
où x_i et y_i représentent l'abscisse et l'ordonnée du point M_i,
et i varie entre 1 et n.

Problème : Trouver les suites de points $M_{i_1}, M_{i_2}, ..., M_{i_p}$,
maximales pour l'inclusion (c'est-à-dire les plus longues), vérifiant :
(1) $x_{i_j} < x_{i_{j+1}}$, pour tout j compris entre 1 et i_p; la suite de points est ordonnée par x croissants;
(2) $y_{i_j} < y_{i_{j+1}}$, pour tout j compris entre 1 et i_p; la suite de points est croissante;
(3) $p_j = \frac{y_{i_j} - y_{i_{j-1}}}{x_{i_j} - x_{i_{j-1}}} < p_{j+1} = \frac{y_{i_{j+1}} - y_{i_j}}{x_{i_{j+1}} - x_{i_j}}$; la pente en j < la pente en j+1; la courbe est supposée convexe.

Algorithme : • 1) trier les points M_i par x_i croissants. Cette étape est au pire (avec un mauvais algorithme de tri) en n^2.

• 2) constituer C, l'ensemble de tous les couples de points $C_{ij} = M_iM_j$ possibles et calculer les pentes correspondantes. On trouve $C_n^2 = n \cdot (n-1)/2$ couples de points possibles. Éliminer tous les couples dont les pentes sont négatives.

• 3) constituer les sous-ensembles C_i' de C, i variant de 1 à n, définis par
$C_i' = \{ M_iM_j$ / i fixé, j varie de i+1 à n$\}$.
trier chaque sous-ensemble de couples de points par pente croissante .
Cette étape est encore en n^2 (au pire le premier sommet voit n-1 autres sommets, le second en voit n-2, etc.), comme dans l'item 1.

• 4) constituer le graphe G défini de la façon suivante : un noeud de G est nécessairement un couple de points, et une arête existe entre deux noeuds $p = M_pM_p'$ et $q = M_qM_q'$de G si le second point de p (M_p') est le premier point de q (M_q) et la pente calculée pour le couple p est inférieure à la pente pour le couple q.

Pour constituer ce graphe, il suffit de rechercher le suivant de chaque noeud dans chaque C_i', et au pire, il y a n-1 noeuds suivants, puisqu'un point a au maximum n-1 suivants. Comme on part de n(n-1) noeuds M_pM_p', possibles pour G, cela nous fait au maximum de l'ordre de n^3 évaluations.

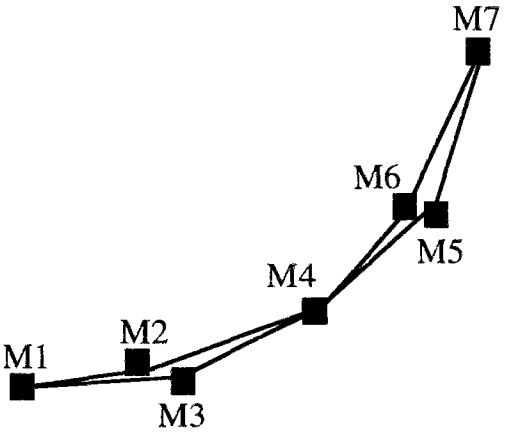
• 5) parcourir ce graphe, en ne retenant que les chemins dont la longueur est maximale à une unité près (longueur = n_{max} ou longueur = n_{max} -1).

Le graphe G est sans circuit par construction. Le plus long chemin existe, et on démontre sans difficulté (voir (Gondran & 84)) que pour le trouver, le nombre d'étapes nécessaires est de l'ordre du nombre d'arêtes. Or, le nombre d'arêtes de ce graphe est d'environ $C_n^3 = \frac{n \cdot (n-1) \cdot (n-2)}{6}$, puisque, au maximum, une arête existe chaque fois qu'on a exhibé 3 sommets parmi les n points du nuage.

Conclusion : les étapes 4 et 5 sont en n^3, les autres sont en n^2. Finalement, l'algorithme est en n^3.

Remarques : - le nombre de chemins possibles est dans certains cas très grand, exponentiel, comme l'exemple de la figure ci-dessous le montre: il faut imaginer que le nuage de points est constitué d'ensemble de points tels que M1, M2, M3, M4, qu'on fait tourner chaque fois d'un très petit angle, et qu'on rajoute chaque fois bout à bout. Supposons qu'on ait répété cette opération k fois. Pour passer du point M1 à M4, on trouve deux chemins convexes et croissants, et au total, on définit pour tout le nuage, 2^k courbes croissantes et convexes de longueur 3*k.

figure 8 : **comment générer un nuage de points où l'on fait passer un nombre exponentiel de courbes croissantes**



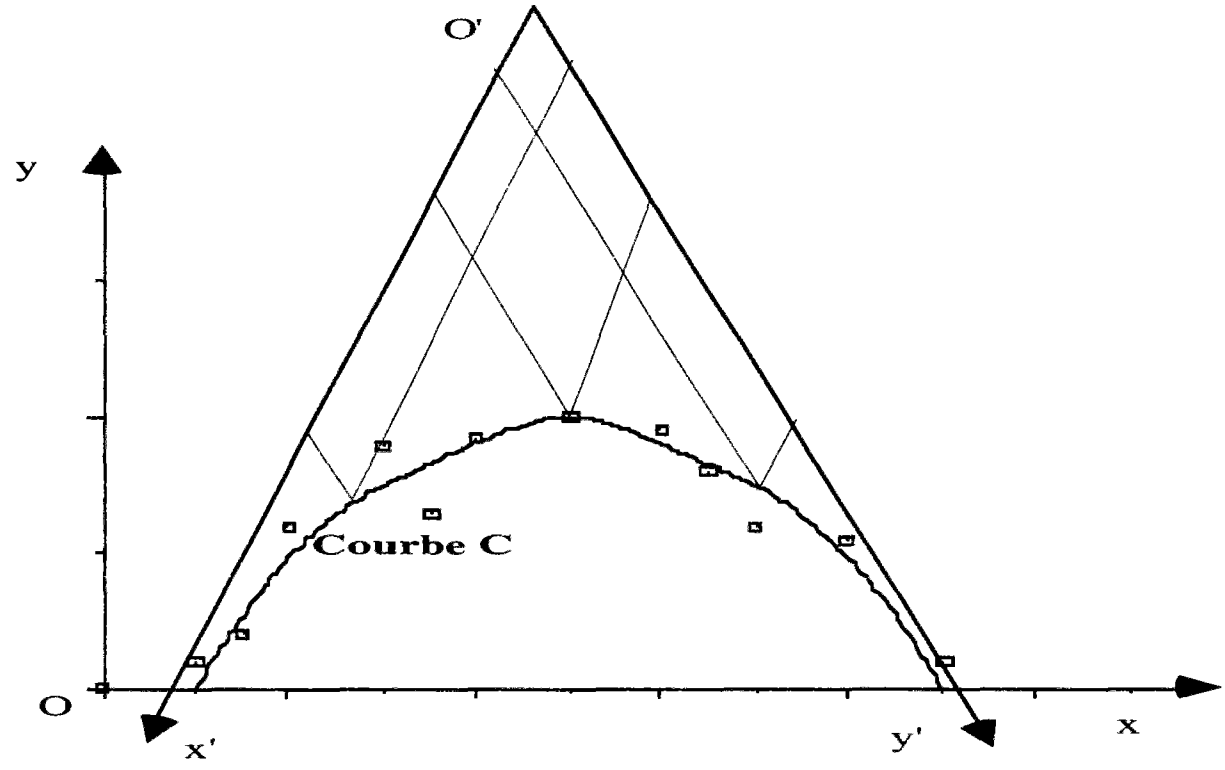
- concrètement, les cas expérimentaux ne nous placent pas dans les plus mauvaises conditions. Il n'est pas invraisemblable, au moment

de programmer cet algorithme, de générer toutes les lignes, sans plus de considérations.
- le fait d'imposer une concavité ne change rien aux démonstrations. L'étape 4 est facultative.

VI.2.3.1.5) Rendre une courbe monotone par changement de repères

Pour tenir compte du cas des courbes à concavité donnée, nous avons mis au point la méthode suivante, qui consiste à se ramener, par un **changement de représentation**, au cas des courbes précédentes, monotones et à concavité donnée. Ici, le changement de représentation est un changement de repère. En effet, une courbe à concavité négative (respectivement positive) peut se transformer par un changement de repère en une courbe décroissante à concavité négative, (croissante à concavité négative) comme l'illustre la figure suivante :

figure 9 : **changement d'axes permettant de rendre la courbe (C) monotone**



Dans le repère O'x'y', la courbe C est devenue monotone décroissante et convexe : plus x' augmente, plus y' diminue (la démonstration formelle de cette proposition, qui n'offre pas de difficulté particulière, est reportée en annexe VI.A.4).

Reste à trouver le repère O'x'y' adéquat, puisque nous disposons d'une seule information à la base : parmi l'ensemble des points du nuage, passe une courbe concave ou convexe. Nous allons nous aider de l'algorithme de génération des lignes, exposé précédemment.

- Nous recherchons toutes les suites de points du nuage qui permettent de définir des lignes concaves (pour se fixer les idées).
- Puis, nous calculons pour chacune de ces lignes la dérivée au premier point de définition, et nous en déduisons la plus grande dérivée (appelons-la D). Nous cherchons alors parmi les parties croissantes des lignes précédentes, le point (P) d'abscisse la plus faible. Nous définissons enfin le premier axe, comme étant la droite qui passe par P et dont la pente vaut D.
- Nous procédons de même pour l'autre axe, en nous appuyant sur la plus grande dérivée calculée au dernier point de définition des lignes, et sur le point le plus extérieur parmi les parties décroissantes des lignes.

Les deux nouveaux axes sont maintenant connus.

Dans ce repère, la courbe recherchée sera devenue monotone.

VI.2.3.1.6) Discussion

Chaque étape de cet algorithme est **simple** conceptuellement et à mettre en oeuvre, ce qui est immédiatement avantageux.

C'est aussi un algorithme **systématique**, sans problèmes d'initialisation, au contraire d'autres méthodes basées sur des résolutions par optimisations d'équations non linéaires. On évite de se poser en fait deux séries de questions :

• d'abord, quelle est la forme de courbes qui convient le mieux (une exponentielle, une hyperbole, ...), qui dépend des cas traités. Il n'est pas interdit de tomber sur des courbes qui se modélisent difficilement par des équations.

• Ensuite, quelles sont les bonnes conditions d'initialisation. Chaque type de courbe conduit à mettre au point une méthode particulière d'identification. Par exemple, il est possible de trouver les coefficients d'un polynôme du troisième degré par une méthode de moindres carrés, tandis que des exponentielles, pour être ajustées, peuvent demander l'emploi de routines d'optimisation, dont la convergence dépend d'une appréciation exacte des valeurs initiales, et ce dernier point peut s'avérer délicat.

La façon de faire, qui aurait consisté à tester systématiquement tous ces types de courbes, pour retenir finalement la meilleure, aurait typiquement présenté ces deux grandes difficultés.

Cet algorithme peut offrir des **variantes** que nous n'avons pas explorées, qui dépendent principalement du critère d'évaluation adopté. En particulier, la norme L^1,

$$\sum_{i=1}^{n} \left| Y_i - (\lambda f(X_i) + \mu) \right|$$

basée sur les valeurs absolues aurait pu convenir. Ce critère a l'avantage de minimiser l'influence des points aberrants, mais aussi l'inconvénient de n'être pas disponible en standard dans les bibliothèques mathématiques spécialisées. La norme L^2, que nous avons adoptée pour notre compte (voir le critère du paragraphe VI.2.3.1.4 est par contre extrêmement facile à mettre en oeuvre.

D'autres méthodes de génération (étape 1 de l'algorithme) et d'élimination (étape 2) sont certainement envisageables (mais nous laisserons ce soin au lecteur). Par contre, changer de méthode d'interpolation (la méthode qui permet à partir de quelques points de définition de tracer les courbes), ne doit pas, à notre avis, amener de grandes différences, à partir du moment où l'interpolation donne des résultats visuels raisonnables, mais cette assertion reste à confirmer. Nous ne répondons de rien, si l'on choisit des interpolations linéaires.

Un autre pas pourrait être franchi pour **reconnaître automatiquement** si la courbe qu'on doit tracer dans un nuage de points, est monotone ou non. Ce soin est actuellement laissé aux experts.

M. Moulet (Moulet 93) a particulièrement étudié cette question dans son travail sur la découverte automatique de lois numériques : pour déterminer si des séries de nombres conduisent à une formule analytique, on recherche avant tout les dépendances monotones entre les données; peut-on considérer que des variables croissent ou décroissent de façon conjointe ou opposée ? Dans ce but, M. Moulet propose l'heuristique suivante : si dans la plupart des cas (70% pour fixer les idées) des variables croissent ensemble (respectivement de façon opposée), à l'incertitude de mesure près, elles sont considérées en dépendance croissante (respectivement décroissante).

Cependant, les statisticiens ont également analysé ce problème, et une première approche de leurs travaux peut être trouvée dans (Lebart & 82).

shéma n° 10 : **algorithme résumé :**
lissage selon l'allure a priori de la courbe

• Génération des lignes qui respectent les critères de monotonie et de concavité donnés

rechercher Nmax = nombre de points de base maximum compris dans une ligne

• Élimination des lignes inadéquates
on ne retient que les lignes telles que :
nombre des points de base de la ligne >= Nmax-1

• Test et ajustement

- Cas des courbes monotones
  Calculer le critère pour chaque ligne retenue
  Retenir la ligne qui rend ce critère minimum
  Calculer la ligne résultante

- Cas des courbes convexes ou concaves
  Trouver la pente maximum (ou minimum) au premier et au dernier point de base
  Faire le changement de repère (les lignes sont devenues de plus monotones dans ce repère)
  Calculer le critère pour chaque ligne
  Retenir la ligne qui rend ce critère minimum
  Calculer la ligne résultante
  Faire le changement de repère inverse

VI.2.3.2) COMPLÉTER DES COURBES A PARTIR DE CAS

Nous allons maintenant aborder une autre nouvelle méthode de tracé des courbes. A la base, nous disposons de plus d'information que précédemment, puisque nous pouvons exploiter une série d'exemples, entre lesquels il est "naturel" d'interpoler. La méthode qui va suivre s'appuie sur un problème concret qui nous a été soumis : il s'agit, comme l'exemple présenté au paragraphe VI.2.2.1.2 l'illustre déjà, de compléter des courbes de distillation; elle se généralise sans peine à des situations analogues.

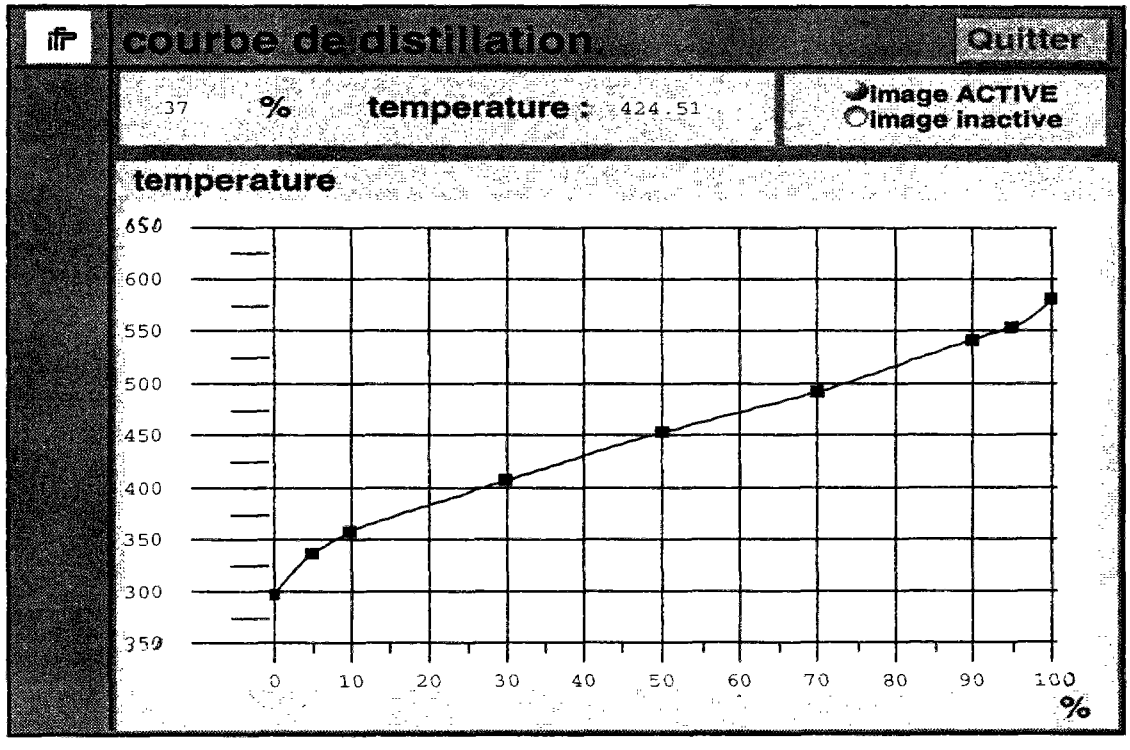
VI.2.3.2.1) Présentation

Pour compléter une courbe de distillation informatiquement, il faut d'abord savoir refaire ce que les ingénieurs font à la main, c'est-à-dire dessiner une courbe d'allure correcte à partir des 9 points, 0%, 5%, 10%, 30%, 50%, 70%, 90%, 95%, 100%. A la main, ces neuf points suffisent pour obtenir un tracé représentatif. En toute rigueur, les normes ASTM établies pour ces mesures (voir chapitre II *contexte et problèmes*) sont définies à partir des 13 points 0% (point initial ou PI), 5%, 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90%, 95%, 100% (point final ou PF).

En réalité, souvent sur un appel d'offre, on ne dispose que de trois points, tels que 0%, 50%, et 100%, ou encore de points inégalement répartis le long de la courbe, tels que 0%, 10%, 30%, 70%, 100%.

Le problème que nous allons traiter ici est de définir complètement une courbe de distillation à partir de ces quelques points

figure 11 : **un exemple de courbe de distillation, tracée à partir de PRINCE**



Tracer une courbe à partir des neuf points cités plus haut, a été résolu en utilisant l'interpolation mise au point par Stineman, développée en annexe du chapitre.

Compléter une courbe à 9 points, quand certains points manquent, s'est révélé un problème beaucoup plus ardu. Les modèles analytiques sont dans cette situation, difficiles à employer. En effet, le nombre de données de départ est variable, et les points sont inégalement répartis. Toutes les variantes possibles de ces conditions de points et de répartition conduisent à imaginer une grande diversité d'algorithmes.

Les équations ont par ailleurs des comportements aux limites parfois difficilement maîtrisables, des tangentes aux points extrêmes imposées par exemple. Voici à titre d'illustration, l'équation imaginée par H. Dhulesia (Dhulesia 84) :

$$V = \{1 - e^{[(\frac{T^*}{\alpha})^{\beta}]}\} * 100,$$

avec $T^* = \frac{T - T_i}{T - T_f}$

où

V est le pourcentage distillé à la température T,
T_i est la température initiale,
T_f la température finale,
a et b sont des constantes positives à déterminer, propres à la coupe considérée.

L'idée développée ici est en revanche extrêmement simple à mettre en oeuvre, quels que soient le nombre de points de départ et leur répartition entre les points PI et PF. Cette idée est également naturelle, puisqu'elle s'appuie sur une analyse de la façon de faire des experts. Enfin, la courbe obtenue passe exactement par les points donnés au départ.

Le principe consiste à partir de courbes existantes. Ces courbes résultent de mesures classiques en laboratoire d'analyse, et sont donc aisément disponibles. Puis, on déforme les courbes pour qu'elles s'adaptent aux données disponibles. Deux courbes de distillation d'un même type de charge se ressemblent visuellement. Il paraît donc naturel de rechercher parmi les exemples de distillation dont on dispose, ceux qui s'approchent le plus du cas à traiter, et de s'en servir comme base pour obtenir un nouveau tracé. Cette méthode soulève à son tour deux difficultés : choisir les courbes de base, savoir comment les déformer.

VI.2.3.2.2) Démarche

La démarche proposée s'inspire d'une méthode d'intelligence artificielle appelée 'case-based reasoning', raisonnement à partir de cas. On retrouve ici les principales étapes de la méthode telle qu'elle est décrite dans la littérature (voir par exemple (Riesbeck & 1989)). Le premier item de la démarche est de retrouver les situations qu'on a mémorisées. Il faut donc les avoir étiquetées avec pertinence, c'est le problème de l'**indexation**. Ayant dégagé des cas possibles, à partir desquels raisonner, on juge leur adéquation, et on se pose des questions de **proximité**. Enfin, l'expert développe des méthodes de déformation pour rechercher une solution possible. C'est la phase d'**adaptation**.

Indexation

Le problème de l'indexation peut être facilement résolu. En effet, la base de cas est constituée pour nous d'une base de charges, analysées à l'IFP au cours des années. Les charges sont stockées par type, et chaque enregistrement contient les valeurs d'un certain nombre de mesures effectuées sur elles, dont les valeurs de la distillation.

Comme les courbes à compléter sont, a priori, d'autant mieux ajustées qu'on part d'exemples proches, pour isoler k courbes qui vont servir de référence, on procède comme suit : si l'on veut obtenir une ASTM D1160 (voir le chapitre II *contexte et problèmes*) pour un distillat sous vide (DSV), le mieux est bien sûr de se baser sur des exemples de distillation ASTM D1160, issues d'analyses de DSV, facilement obtenues en laboratoire spécialisé, disponibles en tout cas à l'IFP. De même pour des D86 sur des gazoles, des TBP sur des 'light cycle oils' (LCO), etc.. Dans chaque cas, une dizaine d'exemples suffisent, qu'on va chercher dans la base de données avec les bons critères.

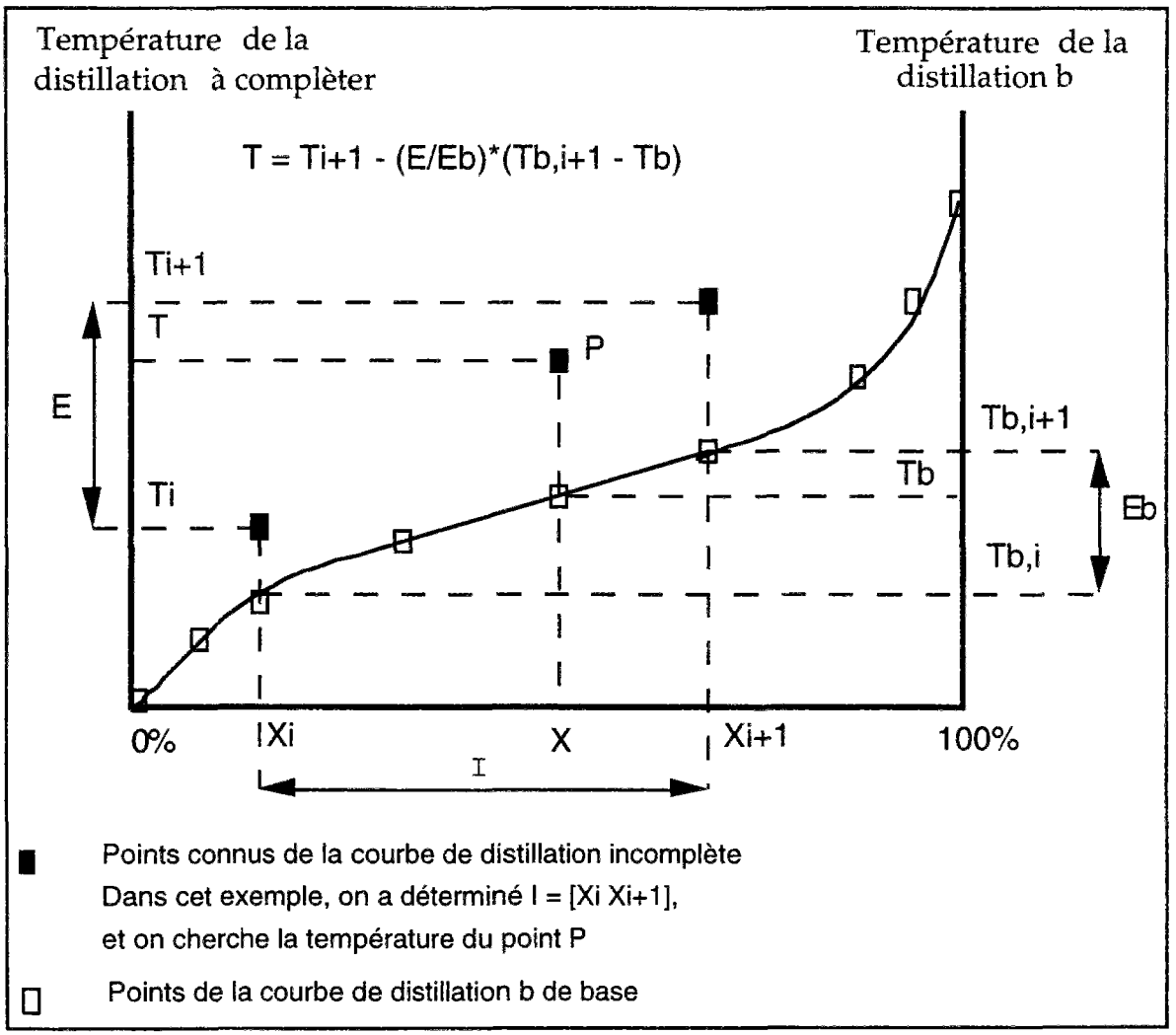
Adaptation (déformation)

Soient maintenant les points (X_i, T_i) représentant les points de base, tels que par exemple les points (X_1= 0%, T_1= 300 °C), (X_2 = 30%, T_2= 400 °C), (X_3 = 70%, T_3= 470 °C) où les pourcentages sont ici des pourcentages volumiques puisqu'il s'agit d'un distillation ASTM D1160 d'une coupe 'distillat sous vide'. L'indice i varie de 1 à n, et nous supposerons bien entendu que n est strictement inférieur à 9 (si n vaut 9, le problème est résolu naturellement par l'interpolation mise en oeuvre). X_i est le pourcentage poids ou volume du point i, et T_i sa température.

Pour calculer la température T correspondant au point P de pourcentage X, il faut d'abord déterminer l'intervalle I = [X_i X_{i+1}] qui encadre le mieux le point P (voir schéma page suivante). Si P se situe avant n'importe quel point P_i de base, on prend I = [X_1 X_2], si P se trouve après tous les points P_i, on prend I = [X_{n-1} X_n]. On définit également la différence E = T_{i+1} - T_i où T_i est la température du point i et T_{i+1} celle du point i+1. Pour reprendre l'exemple précédent [(X_1= 0%, T_1= 300 °C), (X_2 = 30%, T_2= 400 °C), (X_3 = 70%, T_3= 470 °C)]., si l'on cherche à calculer la température du point P d'abscisse X = 50 %, I a pour valeur dans ce cas [X_2 X_3].

Sur chacune des k courbes que l'on a choisies comme bases, on considère les températures $T_{i,k}$ et $T_{i+1,k}$ correspondant aux points d'abscisse X_i et X_{i+1} respectivement, bornes de l'intervalle I précédemment déterminé. On calcule alors tous les écarts $E_k = T_{i+1,k} - T_{i,k}$ pour les k courbes et on compare cette suite d'écarts à $E = T_{i+1} - T_i$. Parmi les courbes k dont l'écart E_k est très proche de l'écart E, on choisit une courbe b qui servira de base pour déterminer le point P. Avant de préciser comment b est choisie, expliquons comment T est calculée quand b est connue.

figure 12 : **Détermination du point (X, T) à partir de la courbe b, dont l'écart E_b est le plus proche de E**



Si le point P devait se situer sur la courbe de distillation b choisie, il aurait pour coordonnées (X, T_b), où T_b est calculée avec l'interpolation de Stineman. Nous prendrons alors :
$T = T_{i+1} - (E/E_b)*(T_{b,i+1} - T_b)$,
c'est-à-dire que nous ramenons l'écart $E_b = T_{b,i+1} - T_{b,i}$ lu sur la courbe b à l'écart E réellement observé.

On itère cette méthode chaque fois qu'il faut trouver un des neuf points manquants. On prend la précaution de réintégrer à chaque passage le point calculé aux points de base.

Proximité

Si l'on s'était contenté de prendre comme base du tracé pour calculer le point P, la courbe dont l'écart E_k était le plus proche de E, on aurait observé qu'une très légère variation des données (un écart de 1°C sur une température d'un des points de base) peut entraîner des changements d'allure nettement perceptibles. Comme les points sont calculés de proche en proche, une petite différence sur l'un des points donnés ou déduits (le point 70% pour fixer les idées) entraîne un écart plus grand sur les points calculés ensuite (le point 90%, puis 95% et 100%).

Par exemple, si l'on part des deux points ASTM D1160 (5%, 377,5 °C) et (70%, 477°C), on obtient de cette façon la courbe complète de la figure 13, tandis qu'en commençant avec les points (5%, 376,5 °C) et (70%, 477 °C), on obtient la figure 14.

figure 13 : **courbe obtenue en partant des points (5%, 377,5 °C) et (70%, 477°C)**

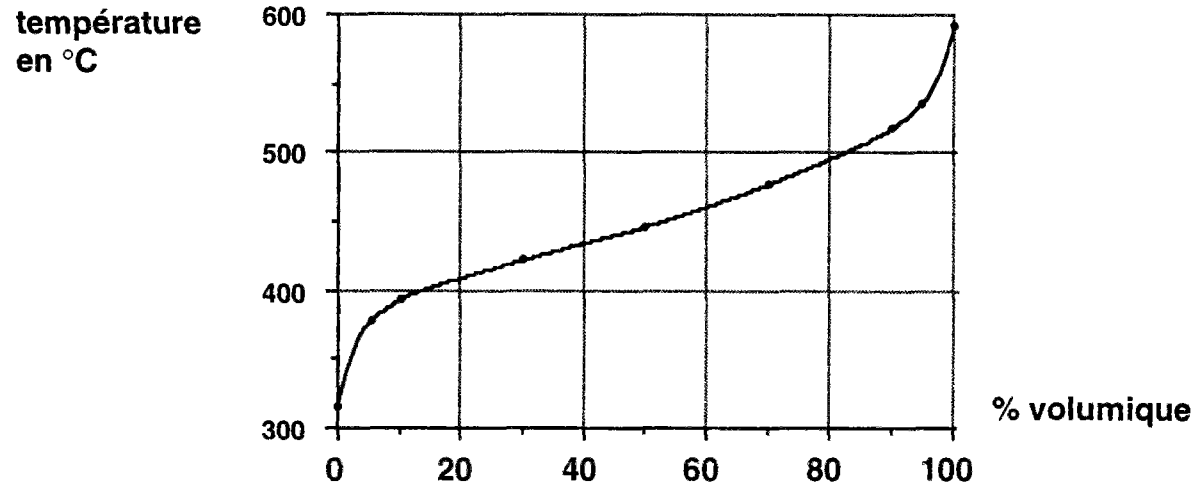
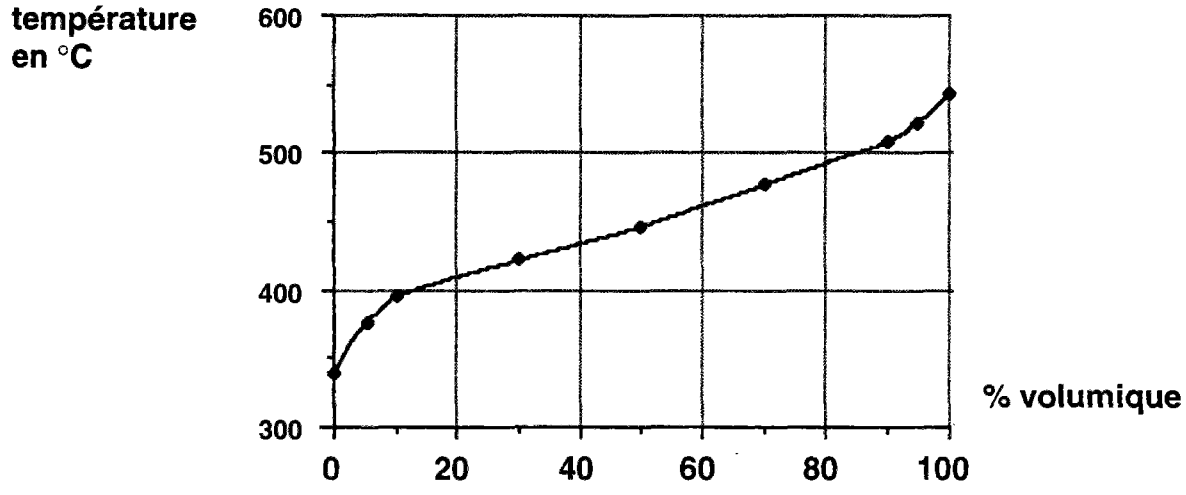


figure 14 : **courbe obtenue en partant des points (5%, 376,5 °C), (70%, 477 °C)**



La liste complète des points des deux courbes est donnée dans le tableau ci-après.

% vol.		0	5	10	30	50	70	90	95	100
Temp. en °C	Fig. 12	316	**377,5**	393	422	446	**477**	518	537	592
	Fig. 13	339	**376,5**	395	422	446	**477**	508	523	544

On observe une différence de 23 °C sur le point 0%, et 48 °C sur le point 100 %. Ceci provient du fait que les courbes de base retenues par cet algorithme ne sont pas les mêmes.

C'est pour éviter ce défaut que la méthode proposée est un peu plus sophistiquée. Concrètement, dans un premier temps, toutes les courbes dont la différence E_k - E est inférieure à une constante e, sont retenues. Cette limite, e, est choisie arbitrairement. Elle dépend de l'unité dans laquelle s'exprime les températures, de l'incertitude des mesures et de la base d'exemples. Avec des températures en degré Celsius et nos exemples de base, nous avons pris e = 2 °C.

Pour déterminer la courbe b base du tracé, il faut un critère de base supplémentaire, qui permette de faire le tri parmi les courbes (C) retenues dans cette première sélection : on suppose implicitement que deux coupes pétrolières proches auront deux courbes de distillation voisines. C'est pourquoi, on considère les coupes dont les courbes restantes (C) sont issues et on use d'un critère de proximité qui s'applique aux coupes plutôt qu'aux distillations elles-mêmes. Ce critère est par exemple la masse volumique (à 15 °C), ou la viscosité (à 50 °C), ou encore une combinaison des deux. Le choix est large. On retient la charge dont, par exemple, la masse volumique est la plus proche de la masse volumique de la charge dont on cherche la distillation.

Bien évidemment, cette méthode donne des résultats d'autant meilleurs que le nombre de points de départ est grand. La courbe de distillation ainsi reconstituée n'est sûrement pas celle qu'on aurait obtenue par mesure. Mais les allures des tracés résultants paraissent toujours vraisemblables. Concrètement, un ingénieur n'aurait pas fait mieux, à main levée.

Finalement cette méthode est capable de fournir des résultats dès qu'on dispose de deux points de base, sans garantie de précision dans ce cas. Il est alors prudent de disposer de deux points situés de part et d'autre du point 50%.

schéma n° 15 :

algorithme résumé : interpolation d'une base de cas

Choisir les k charges de référence

Tant que

il reste un point manquant
parmi les points 0, 5, 10, 30, 50, 70, 90, 95 et 100 %,

- prendre le premier point manquant sur la distillation incomplète;
- trouver les deux points les plus proches et déterminer l'intervalle I, ainsi que l'écart E;
- calculer les écarts Ek, correspondant à I, pour les k courbes;
- sélectionner les charges dont l'écart Ek est très proche de E;
- parmi ces dernières, retenir la charge b la plus voisine (au sens de la masse volumique par exemple);
- calculer la température du point manquant;
- réintégrer ce point dans les points de base;

fin Tant que

VI.3) EXPLOITATION DES COURBES PAR LES EXPERTS

VI.3.1) **VALIDATION**

Le logiciel PRINCE est fortement basé sur les courbes ou corrélations, qui constituent une partie essentielle de la connaissance. C'est ce qui justifie l'intérêt et l'importance des traitements conçus autour de ces notions, pour que l'expert puisse les manipuler comme bon lui semble.

La validation de l'application dépend donc de la validation de ces courbes. Les trois critères habituels de validation des systèmes sont présents dans l'outil mis en place : **la cohérence, la complétude et la pertinence.**

La **cohérence** des connaissances est mesurée par les évolutions des courbes. On valide en particulier les points expérimentaux en y faisant passer des courbes. On met ainsi en évidence les points aberrants et l'expert peut y apporter toutes les mesures de correction qu'il juge nécessaires. De même, une corrélation entre plusieurs variables, issue d'une formule ou d'un programme, peut conduire à des tracés anormaux, que l'expert a tout loisir d'examiner.

En fait, construction de la connaissance et validation sont intimement liées et concomitantes. On valide une corrélation, qu'elle provienne directement des points expérimentaux (les courbes sont définies par les points) ou indirectement par une formule, en analysant les courbes qu'elle produit.

Comme on examine l'évolution des courbes sur tout le domaine de définition, on rend la connaissance plus **complète**, valable sur l'ensemble des valeurs choisies pour chaque variable.

La **pertinence** est assurée parce qu'on part des données expérimentales et non d'une théorie. Les connaissances qu'on en tire sont directement interprétables dans le langage même des experts.

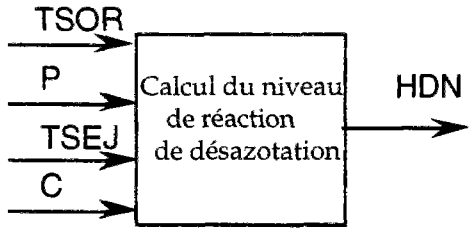
L'expert manipule donc les courbes, que ce soit au moment de la validation ou de la conception, ou encore en utilisation, quand il doit traiter un cas d'étude. Nous allons donc présenter dans cette partie, les différents outils nécessaires pour répondre aux fonctions demandées.

VI.3.2) **UTILISATION**

VI.3.2.1) exemple type

Prenons une exemple type, et considérons le réseau de courbes qui lient les différentes conditions opératoires (Pression (P), Temps de séjour (TSEJ), température 'start of run' (TSOR)), la charge (C) et l'intensité de la réaction de désazotation (HDN, parce que cette réaction est nommée 'HydroDeNitrogenation' en anglais, désazotation à base d'hydrogène). Cet ensemble de courbes permet de calculer, dans sa présentation standard, le paramètre HDN en fonction des autres valeurs, quand elles sont connues. L'exploitation des courbes est la méthode sous-jacente à l'une des inférences qui permettent d'obtenir la valeur du paramètre HDN.

schéma n° 16 : **tâche désazotation**



Les courbes obtenues peuvent être exploitées dans ce cas de deux façons différentes. On peut se demander, comme dans cette présentation, quel est le niveau de désazotation atteint quand on connaît le temps de séjour et la température de réaction. Mais inversement, il est courant de rechercher la température nécessaire qui permet d'atteindre un niveau de désazotation requis, compte tenu d'un temps de séjour imposé, ce qu'on fait graphiquement d'une manière quasiment immédiate : si l'on sait que l'on doit obtenir HDN_0, et qu'il faut travailler à $TSEJ_0$, une interpolation simple entre les différents niveaux de température permet de fixer ce dernier paramètre.

A pression donnée, et pour une charge donnée, quand on affiche ces courbes à l'écran, on obtient un graphique qui a l'allure de la figure 17. Parfois, les utilisateurs veulent connaître l'ensemble des couples temps de séjour-température (TSEJ-TSOR), qui permettent d'atteindre le niveau de performance (HDN) demandé, pour rechercher non plus une valeur optimale, mais un ensemble de réponses possibles, ce que l'on obtient en affichant la courbe de la figure 18.

Et bien évidemment, ce qui est vrai pour un paramètre est vrai pour les autres. En fait ces graphiques sont très riches en informations, qu'il est très difficile ou fastidieux d'extraire manuellement.

figure 17

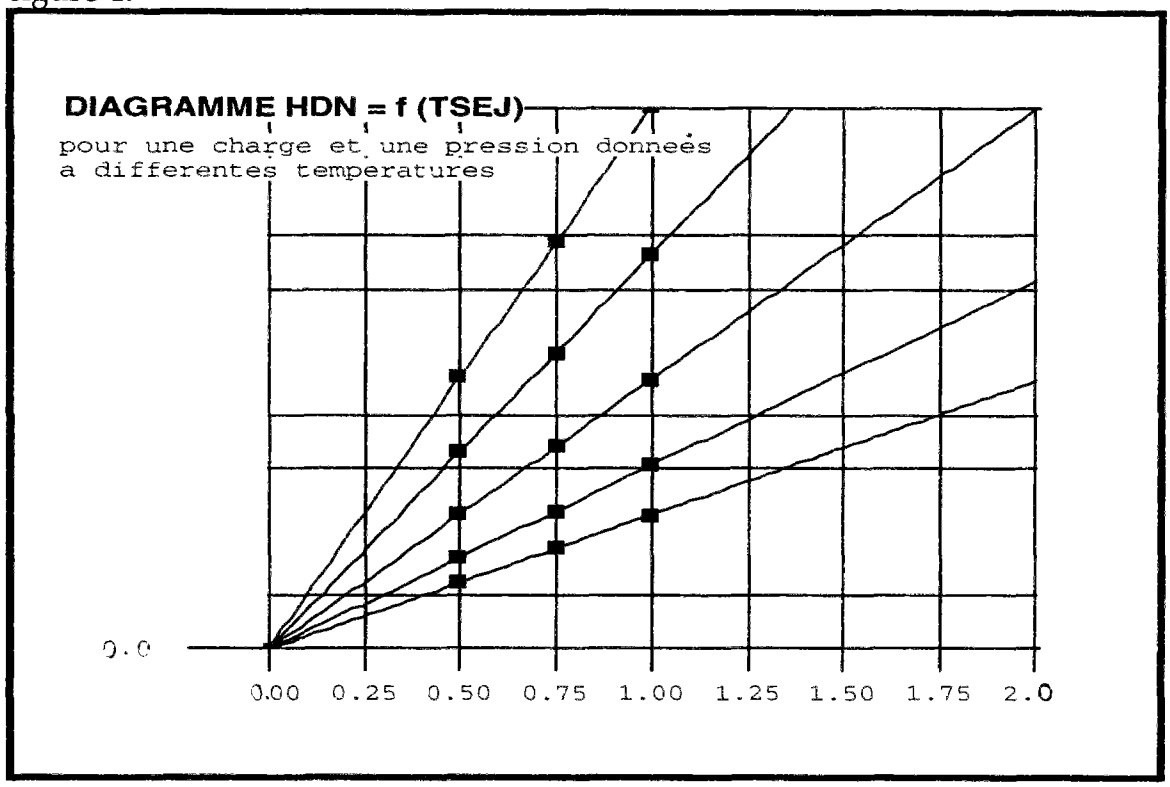
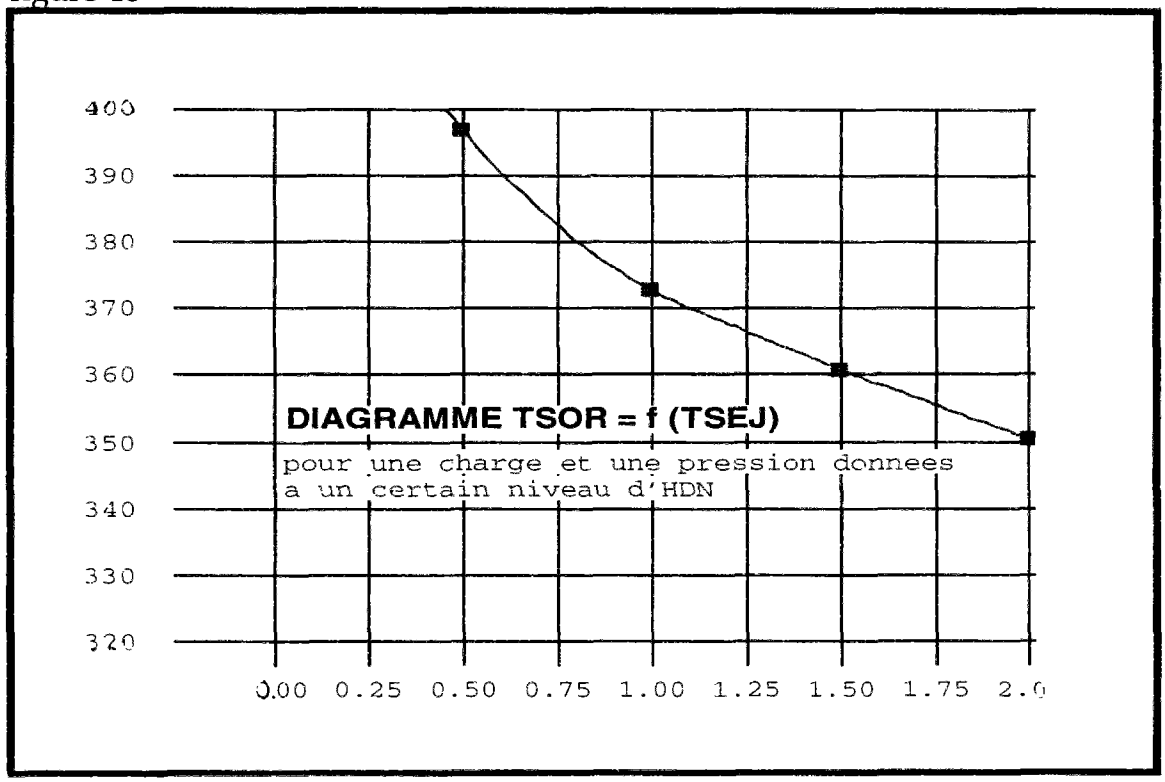


figure 18



VI.3.2.2) Des relations inversibles

Il faut remarquer que les relations qui lient les différentes variables du procédé, sont bien souvent (mais pas toujours) localement inversibles, ce qui signifie, que si la relation est à n variables, la donnée de n-1 d'entre elles, quelles que soient ces n-1 paramètres choisis, permet de trouver, sans aucune indétermination, la variable manquante.

Cette propriété résulte de monotonies locales. Plus précisément, quels que soient p et q, deux indices compris entre 1 et n, les projections de la relation qui lie les variables sur les domaines de définition des variables x_p et x_q, D_p*D_q, sont des fonctions monotones.

C'est cette propriété d'inversion locale qui permet le raisonnement des experts. En conception de procédé, il n'est pas rare d'entendre des raisonnements du type : *si la pression augmente, alors la sévérité du traitement croit également,* et inversement, *si le niveau de performance doit s'accroître, alors il faut augmenter la pression, toutes les autres conditions étant égales par ailleurs.*

VI.3.2.3) DES RELATIONS DE DIFFÉRENTES NATURES ENTRE LES VARIABLES

Les relations qui lient les différentes variables du procédé sont de différentes sortes et donnent lieu à différents algorithmes d'inversion :

• **cas d'un réseau de points**
Il n'existe pas de programme ni de formule pour calculer le résultat attendu, mais un réseau de points de définition sur lequel les différentes courbes sont définies. Ce réseau de points doit être régulier (ou cubique), c'est-à-dire qu'il doit être défini comme le produit cartésien de $I_1*I_2*...*I_{n-1}$, où chaque I est un échantillon de points pris dans les domaines de définition de la variable correspondante.

Si nous souhaitons inverser de tels abaques, nous opérons comme suit :

supposons que le réseau de points nous donne x_4, quand on connaît x_1, x_2, x_3, . Nous souhaitons tracer la courbe $x_3 = f(x_2)$, étant donnés $x_1= X_1$, $x_4 = X_4$.
- Pour cela, nous commençons par calculer les courbes obtenues quand x_1 est fixé à sa valeur de X_1. Plus précisément, la lecture de l'ensemble des points nous permet d'obtenir des courbes x_4 fonction de x_1, pour chaque valeur de x_2 et de x_3 prises dans leur échantillon respectif. En fixant $x_1= X_1$, on calcule alors un ensemble de points reliant x_2, x_3 et x_4.
- Puis, pour chaque valeur de x_2 appartenant à I_2, nous formons la courbe liant x_3 et x_4.
- Nous en déduisons par interpolation (ou extrapolation, si nécessaire) la valeur qu'il faut donner à x_3 pour obtenir X_4. Comme ce calcul a été effectué pour chaque point du domaine I_2, nous obtenons ainsi la courbe désirée, liant x2 et x3.

• **cas d'une formule analytique.**
Il peut exister une formule analytique entre les paramètres du procédé, par exemple :

$$\frac{C_0}{C} = \left[1+k*(n-1)*C_0^{(n-1)}*t\right]^{\frac{1}{(n-1)}}$$

Dans ce cas, le calcul de la courbe consiste à calculer les points nécessaires pour chaque valeur des variables, prises dans leur échantillon.

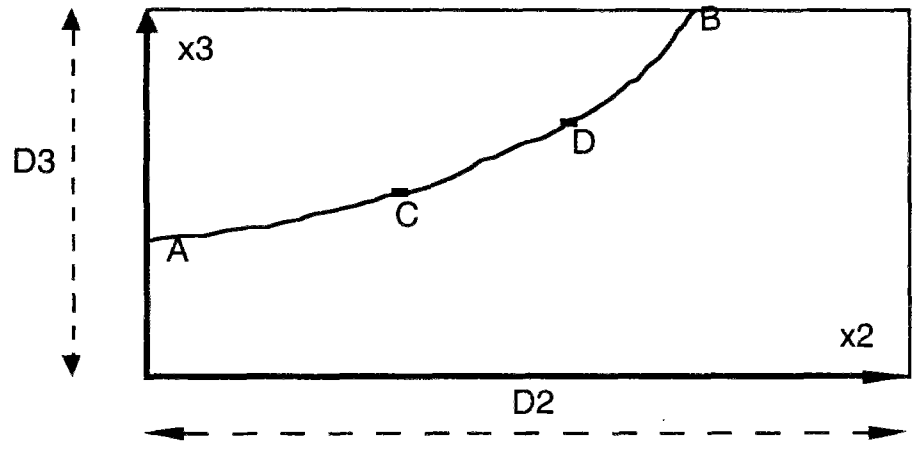
Cette formule peut ou non s'inverser analytiquement. C'est le cas quand la variable par rapport à laquelle nous cherchons à inverser la formule apparaît une seule fois dans celle-ci (comme dans l'exemple). Sous cette condition, un algorithme de calcul formel extrêmement simple permet d'effectuer cette inversion. Si ce n'est pas le cas, nous ne cherchons pas de formule inverse : ceci pourrait s'avérer particulièrement complexe, des logiciels sont consacrés au traitement formel des expressions. Plus simplement, nous nous ramenons au cas d'inversion des programmes, que nous décrivons dans l'item suivant.

• **cas des programmes**
On peut disposer d'un programme, pour calculer un paramètre de sortie. L'affichage direct est extrêmement simple, il suffit de faire tourner le programme autant de fois que nécessaire, comme dans le cas des formules analytiques.

Ce programme peut être la méthode sous-jacente d'une relation inversible, c'est-à-dire, qu'il est possible d'inverser ce programme. Comme il est hors de question de le faire formellement, nous procédons numériquement de la façon suivante :

supposons que le programme nous donne x_4, quand on connaît x_1, x_2, x_3. Nous voulons tracer la courbe liant x_2, x_3 avec $x_1 = X_1$, $x_4 = X_4$, donnés.
- Pour cela, nous commençons par calculer les couples de points x_2, x_3 aux bornes de leur domaine de définition : nous déterminons les points où la courbe cherchée rentre et sort du domaine de définition (aux points A et B sur la figure). En effet, comme la courbe qui relie x_2, x_3 est monotone, elle doit nécessairement couper le domaine D_2*D_3, sur au moins deux côtés

figure 19 : **inversion d'une courbe obtenue par programme**



Pour cela,

- On calcule la valeur de x_4 aux quatre coins du domaine.
- On sélectionne les côtés pour lesquels la valeur X_4 est comprise entre les valeurs de x_4 aux extrémités.
- Cela nous donne deux côtés, puisque la courbe liant x_2, x_3 est monotone. Sur chacun d'eux, une variable est fixée à sa valeur limite, et l'autre est libre. Puis, sur chaque côté choisi, $x_1 = X_1$ étant fixé, par itération sur la variable libre entre les valeurs calculées aux sommets, on calcule le point où la courbe rentre ou sort du domaine. Cela nous donne deux points (A et B, sur la figure).

- Ensuite, il suffit de déterminer un, deux ou trois points intermédiaires (C et D dans notre exemple), et on trace finalement la courbe obtenue, à partir des points ainsi déterminés (les quatre points A, B , C et D, dans notre exemple).

Cet algorithme pourrait être probablement étendu aux domaines de définition convexes.

VI.3.3) MISE EN OEUVRE

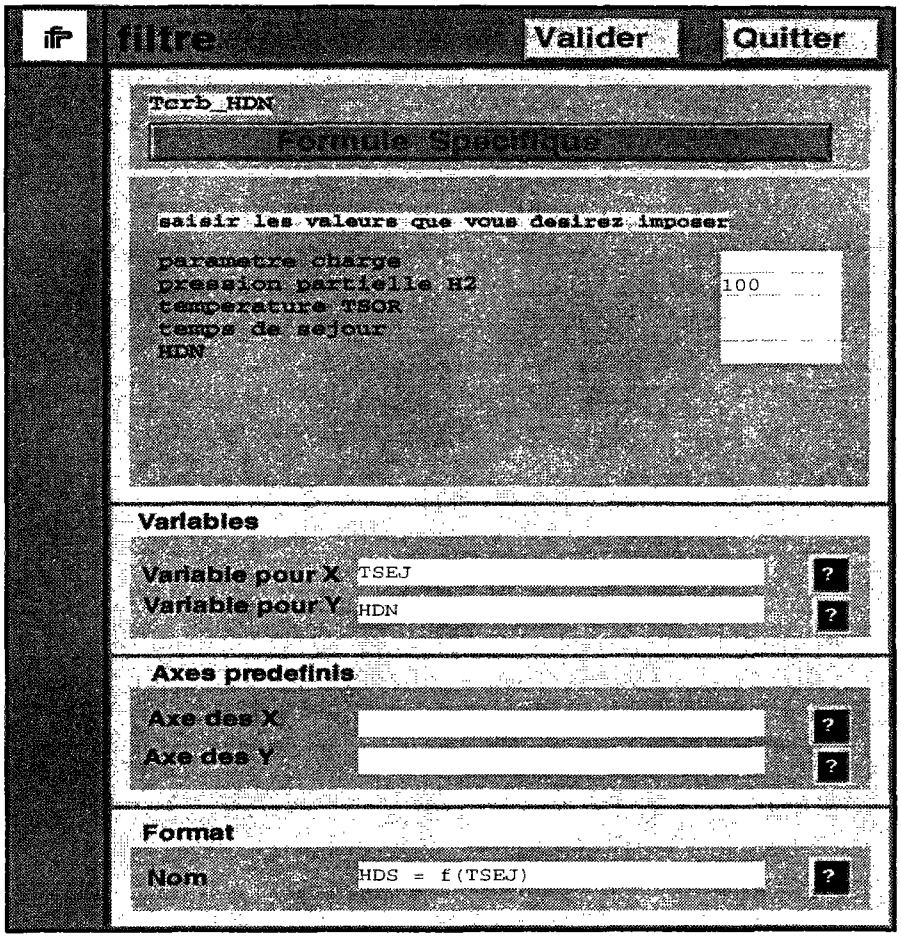
Les écrans qui permettent la visualisation des courbes s'appuient sur trois aspects :
- le découpage en tâches de l'application PRINCE,
- la visualisation des réponses possibles,
- l'analyse des points expérimentaux.

VI.3.3.1) TACHES ET VISUALISATIONS

Comme nous l'avons déjà souligné, chaque pas de raisonnement (qu'on a appelé inférence), quand il est mis en oeuvre dans un problème, devient une tâche. Une inférence se caractérise par la donnée de ses entrées et sorties. La liaison entre les entrées et les sorties peut donner lieu, dans certains cas décidés par les experts, à des visualisations, sous forme de courbes.

Du fait de cette représentation, l'écran de lancement des visualisations est générique. Il comprend une partie variable, qui dépend des courbes que l'on souhaite visualiser, et une partie fixe.

figure 20 : **écran de lancement d'une visualisation**



Dans l'exemple, on souhaite obtenir un graphique d'abscisse TSEJ et d'ordonnée HDN, et ceci pour une pression partielle de 100 (bars), et ceci pour toutes les valeurs des charges et des températures définies.

Sur la partie fixe, on saisit essentiellement le format de visualisation. Plus précisément, si on connaît l'échelle de l'axe des X et (ou) celle de l'axe des Y, des zones sont prévues à cet effet, pour rentrer ces informations. Si de plus, on souhaite réutiliser un format prédéfini dans lequel l'axe des X et celui des Y sont connus, et où des libellés ont été préalablement introduits, on remplit la zone format.

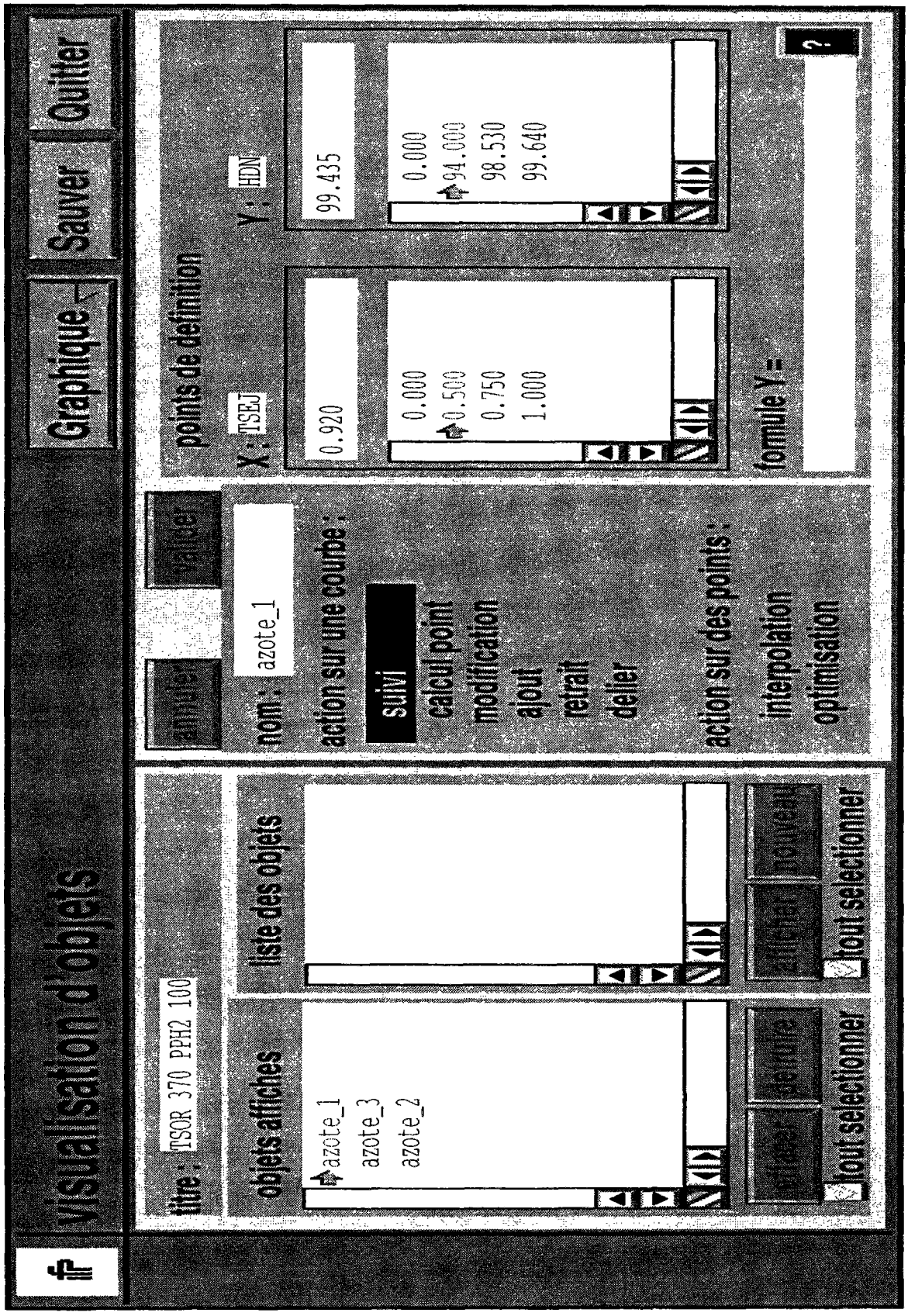
La partie dépendante sert à préciser quelles sont les paramètres imposés de la visualisation. Par exemple, considérons la relation qui lie la charge, les conditions opératoires (pression, température et temps de séjour) et le niveau de la réaction de désazotation (HDN), que nous avons déjà présentée. L'utilisateur peut, s'il le souhaite, visualiser les courbes pour une pression de 100 bars sur les charges de référence. Au contraire, il peut préciser la charge, laisser libre le paramètre de pression, et imposer une valeur pour le temps de séjour. Dans ce cas-là, la seule possibilité qui lui reste est de faire apparaître la courbe représentant la variation de l'HDN en fonction de la température.

Par ailleurs, il est possible dans notre exemple de fixer le niveau de désazotation (et de laisser libre dans ce cas un autre paramètre). Pour cela, nous utiliserions les possibilités d'inversion que nous venons de décrire.

En fait, plutôt que de stocker des inférences, ce sont des relations entre les variables qui sont représentées. Si (x_1, x_2, x_3, x_4, x_5) est une relation entre les cinq variables x_1, x_2, x_3, x_4 et x_5, nous pouvons définir cinq inférences, qui sont : (x_1, x_2, x_3, x_4)--> x_5, (x_1, x_2, x_3, x_5)--> x_4, ..., (x_5, x_2, x_3, x_4)--> x_1. La donnée de l'inconnue permet immédiatement de savoir si c'est un calcul direct qu'il faut effectuer, ou si au contraire, il faut inverser la relation. Comme le calcul direct et le calcul inverse sont très proches, (c'est-à-dire que le calcul inverse utilise abondamment la relation directe), nous n'avons utilisé finalement qu'une seule méthode, attachée à la relation, qui sait s'orienter en fonction des cas présentés.

VI.3.3.2) VISUALISATION DES RÉPONSES POSSIBLES

figure 21 : **écran de définition et de commande des courbes affichées**



Le premier travail est de lire le fichier de points. Deux modes de lecture des points sont possibles : tous les points lus sont indépendants, ou les points lus sur le fichier servent à définir des courbes.

Dans le premier cas, l'écran précédent permet seulement de filtrer les valeurs et de définir les variables et les axes.

Dans le second cas, les points qui peuvent être reliés dans les axes de coordonnées choisis sont regroupés et permettent de tracer des courbes. S'il n'y a que deux points, nous obtenons des droites. Avec plus de points correctement disposés, la méthode d'interpolation choisie, l'interpolation de Stineman, conduit à des courbes.

Après le lancement, et après un certain temps de traitement, les courbes souhaitées apparaissent à l'écran, ainsi qu'une fenêtre de commande sur les courbes affichées. Nous présentons ici, une première série d'actions possibles sur les courbes.

Signalons d'abord la possibilité d'**ajouter** ou de **retirer** des courbes de l'affichage. A l'issue du calcul, on obtient un certain nombre de courbes, et c'est l'utilisateur qui sélectionne celles qu'il veut voir apparaître à l'écran. S'il veut en visualiser d'autres, il efface celles qui sont présentes et choisit les nouvelles.

Deux fonctionnalités sont particulièrement utiles pour l'analyse des courbes :
• la première est le **suivi**, qui permet par un simple "clic" de la souris, de voir dans la zone prévue à cet effet les coordonnées du point choisi sur la courbe sélectionnée.
• comme il est parfois difficile d'obtenir des valeurs exactes, l'utilisateur peut utiliser le bouton '**calcul points**'. Pour cela, soit il fournit la valeur de X et en suivant la courbe, la machine calculera la valeur de Y, soit dans le cas des courbes monotones, il donne la valeur de Y et la machine restitue la valeur de Y.

VI.3.3.3) CAS DES COURBES EXPÉRIMENTALES

Pour analyser les points, nous offrons des facilités supplémentaires. L'idée est de pouvoir agir doublement sur les courbes, par l'intermédiaire de l'affichage ou en modifiant directement les points.

On peut **ajouter un point** à une courbe, soit directement en cliquant sur le graphique, soit indirectement en fournissant les coordonnées du nouveau point. De même, on peut **retirer un point** d'une courbe en le sélectionnant par la souris sur la visualisation, ou en sélectionnant ses coordonnées dans la liste des points de la courbe. On peut également **modifier une courbe**, de la même façon que précédemment. On attrape le point avec la souris, ou on modifie ses coordonnées.

Dans tous les cas, la courbe se retrace immédiatement en fonction des nouvelles informations.

Si l'utilisateur le souhaite, il a la possibilité de rajouter de **nouvelles courbes** à l'écran en fournissant les points de définition de celle-ci, ou en donnant une formule, liant l'abscisse à l'ordonnée. Inversement, évidemment, il peut **détruire** une courbe.

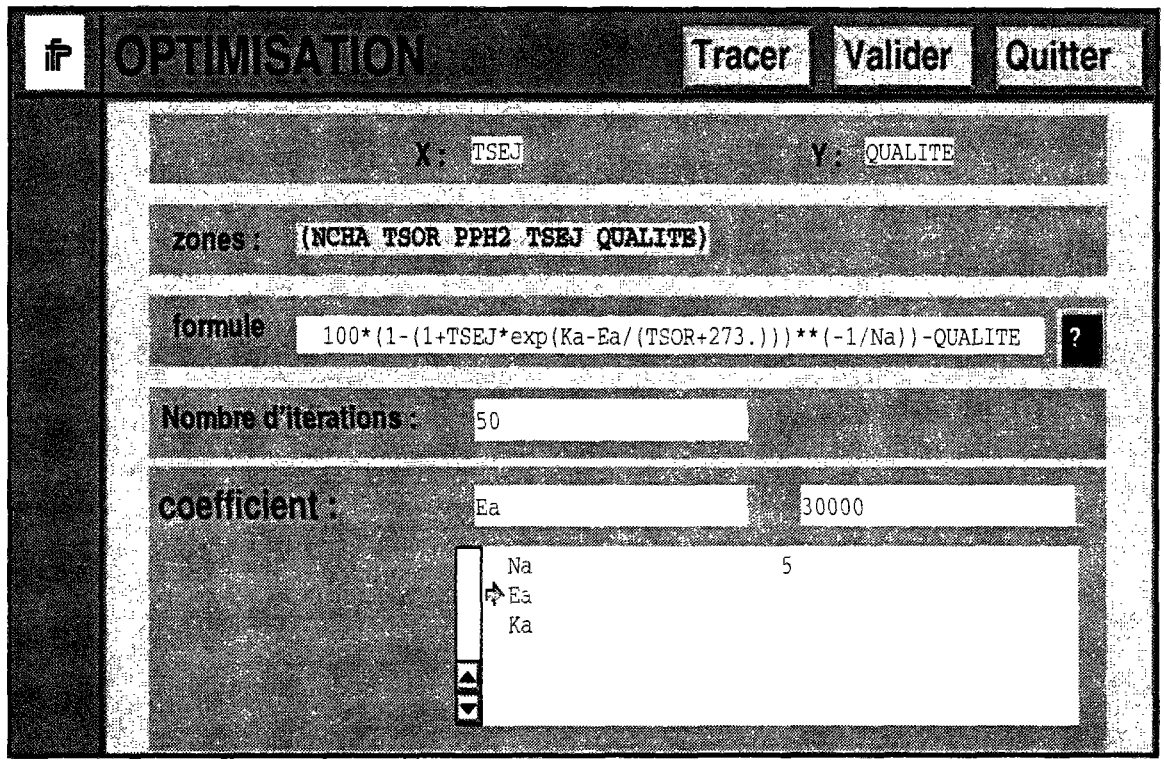
Il est possible de récupérer les points d'une courbe, par l'action du bouton '**délier**' et inversement de faire passer une courbe par des points par le bouton '**interpoler**'.

Enfin, quand on a récupéré des points, souvent on souhaite essayer une formule. C'est l'action du bouton '**optimiser**', puisqu'il s'agit de chercher les bons paramètres d'une formule, dont nous présentons l'écran maintenant.

VI.3.3.4) OPTIMISATION

L'écran permet la saisie d'une formule, le choix des paramètres initiaux de la formule, et le dessin des courbes obtenues.

figure 22 : **contrôle d'une optimisation**



Pour saisir une formule, le plus simple est de passer par l'intermédiaire du bouton 'point d'interrogation', qui donne un exemple de formules possibles. L'écran de la figure 22 donne un exemple d'une formule du domaine de la cinétique. En fait ces formules sont lues dans un petit fichier d'aide qu'il est très facile de modifier pour passer d'un domaine à un autre.

Une fois la formule entrée, le logiciel reconnaît les variables des paramètres, et pour ces derniers, il demande à l'utilisateur de donner des valeurs initiales, ce qu'on ne sait malheureusement pas faire automatiquement aujourd'hui. Une description plus fine de notre façon de faire est décrite en annexe 'ROUTINE D'OPTIMISATION'.

Si l'algorithme d'optimisation a convergé, il rend la valeur des paramètres obtenus. On peut alors tracer les courbes obtenues par l'intermédiaire du bouton adéquat, dans le cas où la variable représentant Y n'apparaît qu'une seule fois dans la formule à optimiser. Les courbes apparaissent directement à l'écran, ce qui permet à l'utilisateur de juger si l'optimisation le satisfait.

4) CONCLUSION

Dans ce chapitre, nous avons présenté les outils mis au point pour que les utilisateurs puissent, d'une part construire de nouvelles corrélations, et d'autre part valider et utiliser celles qu'ils ont déjà développées.

Dans la partie construction de la connaissance et exploitation des points expérimentaux, nous avons proposé des méthodes innovantes pour tracer des courbes, en partant des mêmes informations de base que les experts. La première consiste à tracer des courbes de monotonie et de concavité donnée dans un nuage de points, la seconde permet de tracer des courbes d'une forme donnée comme les courbes de distillation.

Cependant, il est à noter que l'on est encore loin de savoir construire de nouvelles connaissances sans l'intervention des experts utilisateurs. Pour notre part, nous pensons que dans ce domaine, il s'agit de proposer des outils dont l'utilisateur garde la maîtrise, qui l'aide à réaliser ses idées et à valider les connaissances.

Il faut remarquer que l'outillage de base, pour pouvoir s'occuper des courbes, est conséquent, ce qui conduit à développer un grand nombre d'algorithmes différents. Les corrélations graphiques, pétrolières notamment, sont très variées, puisqu'elles s'expriment sous forme d'abaques, d'expressions mathématiques entre des variables ou sous forme de véritables programmes. Nous avons mis en place les moyens nécessaires pour exploiter ces différentes relations, en particulier quand elles sont inversibles

Dans la partie mise en oeuvre, l'accent a été mis sur une interaction double entre la partie graphique, et la partie définition des courbes : on peut agir sur ces dernières, à partir de leur définition, en rentrant les coordonnées d'un nouveau point par exemple, ou à partir de la visualisation, en pointant directement avec la souris sur l'écran.

ANNEXES

VI.A.1) LA MODÉLISATION DES RÉACTIONS CHIMIQUES ET LA LOI D'ARRHENIUS

La théorie cinétique conduit pour une réaction donnée à poser l'**équation cinétique** :

$$(1)\quad -\frac{dC}{dt} = k*C^n$$

t est le temps de réaction,
C est la concentration du réactif,
n est l'ordre de la réaction,
k est la constante cinétique de la réaction.

La thermodynamique définit l'énergie d'activation pour un seul corps par la relation :

$$(2)\quad E = R*T^2*\left(\frac{\partial \ln(k)}{\partial T}\right)_P$$

T est la température absolue,
R est la constante des gaz parfaits,
lorsque P (la pression) est constante.

(voir Chemical Kinetics de Laidler (1965)).

On observe expérimentalement que, à pression donnée, E est constant.
Par intégration, à pression donnée, on retrouve la **loi d'Arrhenius**, observée empiriquement :

$$(3)\ k = k0*e^{\left(-\frac{E}{R*T}\right)}$$

où k0 est la constante d'action ou le facteur de fréquence.

Par intégration de l'équation (1), quand n = 1, on obtient :

$$(1\text{-bis})\ -\ln\left(\frac{C}{C_0}\right) = k'*t$$

et lorsque n est différent de 1

$$(1\text{-ter})\quad \frac{C_0}{C} = \left[1+k*(n-1)*C_0^{(n-1)}*t\right]^{\frac{1}{(n-1)}}$$

où C_0 est la concentration à l'instant 0, et k la constante cinétique.

Application : sur une unité pilote de désulfuration, le soufre S_0 de la charge et de l'effluent S sont mesurés; dans un premier temps, S et C sont assimilés, c'est-à-dire qu'on considère qu'une seule réaction permet d'enlever le soufre. La courbe S en fonction du temps t est tracée, puis par un calcul d'optimisation, les coefficients k et n sont ajustés. L'expérience prouve qu'on y arrive pour une charge donnée. En remplaçant k par son expression tirée de (3), on parvient également à représenter les effets de la température. Par contre l'influence de la pression est beaucoup moins aisée à modéliser, sans parler de l'influence de la charge.

Ce type de calculs a été demandé au logiciel PRINCE. Ils permettent de dépouiller les expériences pilotes effectuées, ainsi que de prédire les résultats dans d'autres conditions de fonctionnement (changement de température, de temps t de réaction, etc.).

VI.A.2) ROUTINE D'OPTIMISATION

La formule

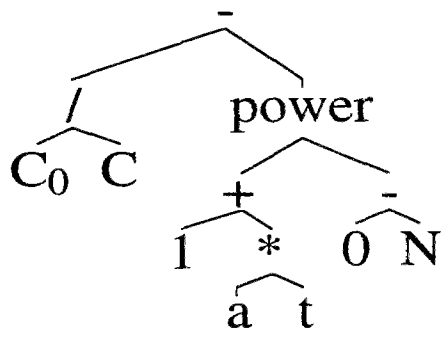
$$(1.4) \quad \frac{C_0}{C} - \left[1+a*t\right]^{-N}$$

est écrite sous la forme habituelle, dite infixée. Cette formule est tirée de l'équation (1-ter) développée à propos de la modélisation des réactions chimiques. Si nous la transformons comme dans le langage LISP en forme préfixée (tous les opérateurs sont en tête de l'expression), cette expression devient :

```
(- (/ (C0 C)
   (power (+ 1     (* a  t))
          (- 0     N)))
```

Il est alors possible de la réécrire sous forme d'un arbre, structure classique qui permet d'évaluer des expressions. On obtient :

schéma n°23 : **expression sous forme d'arbre**



Une forme de ce type est alors très facile à manipuler en langage C. On peut facilement l'évaluer en utilisant les mécanismes de récursivité du C et calculer ses dérivées par rapport à n'importe quelle variable, ce dont nous avons besoin pour notre programme d'optimisation.

En effet, les routines d'optimisation prennent classiquement comme arguments un tableau de points sur lesquels il faut évaluer une expression, un ensemble de paramètres à optimiser, pour lesquels on fournit une valeur initiale et une routine de calcul, qui permet de calculer la valeur de l'expression et de ses dérivées en un point quelconque. C'est cette dernière facilité que nous avons détaillée ici, de telle sorte que nos utilisateurs n'aient plus à écrire une seule ligne de programmation, pour décrire la fonction à évaluer.

La routine d'optimisation est tirée d'une bibliothèque spécialisée et elle a été réécrite en langage C et adaptée à nos fins. La routine a pour tâche de minimiser la somme des carrés des écarts entre la valeur des points et l'expression que l'utilisateur fournit. Si nous reprenons l'exemple précédent, il faut minimiser

$$\sum_{\text{points}} \left(\frac{C_0}{C} - \left[1+a*t\right]^{-N} \right)^2$$

c'est-à-dire trouver les valeurs de a et N qui rendent ce critère le plus petit possible, quand on connaît un ensemble de points repérés par leur coordonnée C, C_0 et t .

Finalement, l'utilisateur saisit une formule sous sa forme habituelle (infixée), saisit les points pour lequel il veut une optimisation, et lance le calcul après avoir fourni les valeurs initiales. L'algorithme d'optimisation choisi semble à première vue suffisamment robuste, pour converger dans des cas normaux. S'il ne converge pas, il le signale et l'utilisateur peut corriger les valeurs initiales fournies. Enfin, comme nous maîtrisons complètement les mécanismes d'évaluation, aucune erreur majeure ne peut survenir, qui fait classiquement stopper ce genre de mécanisme (exemple : 'exponential overflow') : nous testons les cas dangereux, avant leur évaluation, pour pouvoir rendre la main à l'utilisateur, en cas de problème.

VI.A.3) INTERPOLATION

VI.A.3.1) L'ALGORITHME DE STINEMAN

La méthode mise au point par Stineman (Stineman 80), ingénieur chez Boeing, vise à tracer des courbes telles que celles de la figure 24. Si l'on s'y prend brutalement, par exemple en utilisant un polynôme de degré 4, on obtient la figure 25.

figure n°24 : **interpolation de Stineman**

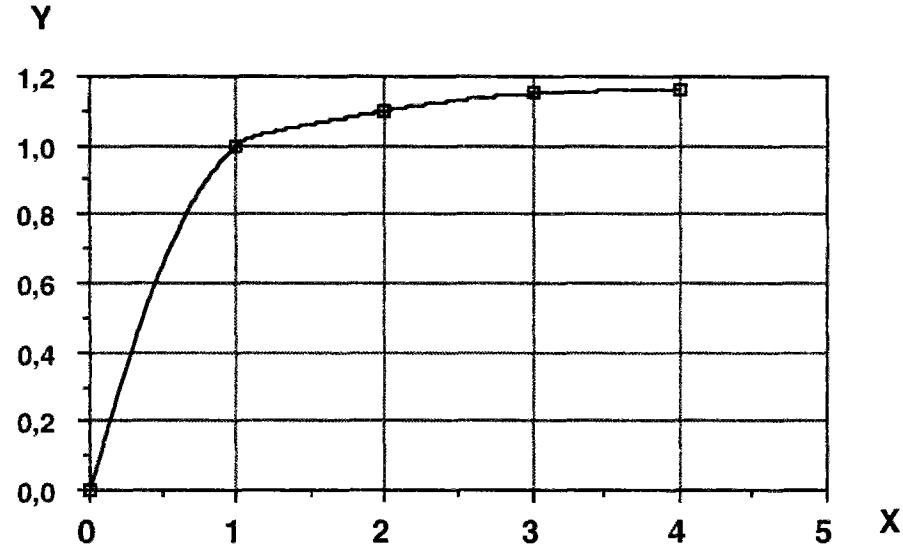
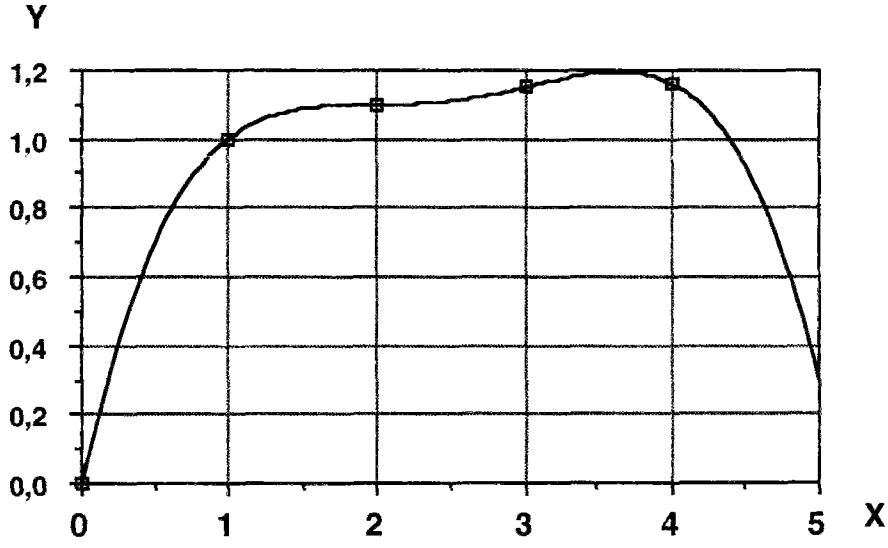


figure n°25 : **courbe polynômiale de degré 4**



La méthode de Stineman suppose au préalable de calculer les tangentes. Quelques cas sont à prévoir :

- Soient I, J et K, trois points consécutifs. Pour estimer la pente de la courbe au point J, on trace le cercle qui passe par les points I, J et K, et la tangente de la courbe au point J sera prise égale à la tangente du cercle en ce point, que J soit au-dessus ou au-dessous de la ligne IJ.

- Si J est le dernier point et I le point qui le précède, on considère la ligne IJ, et la tangente en I, de pente P_i. Si la pente de la ligne IJ est plus grande que P_i , alors on considère que la tangente en J est celle de la parabole qui passe par I et J, et de tangente P_i en I. On sera obligatoirement sur la branche droite d'une parabole à concavité positive.

- Dans le cas contraire, Stineman propose un calcul qui permet de pondérer la pente de la parabole, mais il est plus simple d'inverser les axes et de considérer une parabole d'équation $x = a*y^2+b*y+c$, et de reconduire alors le raisonnement précédent, c'est l'amélioration que nous avons apportée. On choisit dans ce cas la branche supérieure de la parabole.

- Si J est le premier point, on choisit la branche gauche (respectivement basse) de la parabole.

Cette façon de calculer les tangentes aux points terminaux a l'énorme avantage de permettre des extrapolations. En effet, il se trouve que cette manière de procéder correspond exactement à celle des ingénieurs, quand sur papier millimétré, ils prolongent des courbes. Mais ces extrapolations sont sans garantie d'exactitude.

Maintenant que l'on connaît les pentes, en tous les points, on peut calculer les courbes. Deux cas sont possibles :
Soient I et J, deux points consécutifs.

- S'il n'y a pas de point d'inflexion entre I et J, c'est-à-dire si la pente de la tangente en I est plus petite (respectivement plus grande) que la pente de la ligne IJ, qui est elle-même plus petite (respectivement plus grande) que la pente de la tangente en J , on obtient y par l'équation :

$$y=y_0+\frac{\Delta y_j \Delta y_{j+1}}{\Delta y_j+\Delta y_{j+1}}$$

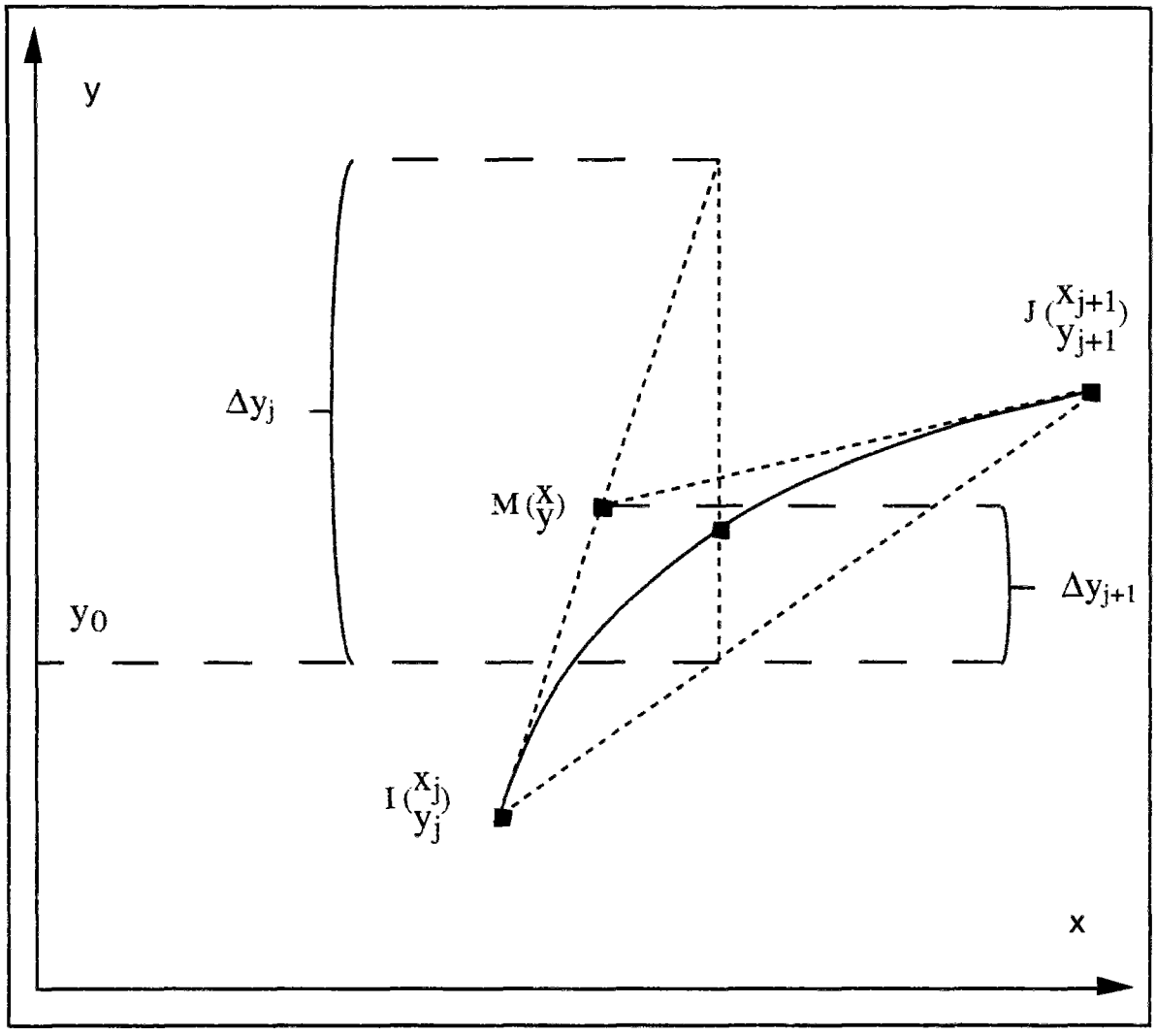
où

$$\Delta y_j=y_j+y'_j(x-x_j)-y_0 \quad \text{et} \quad \Delta y_{j+1}=y_{j+1}+y'_{j+1}(x-x_{j+1})-y_0$$

et

$$y_0=y_j+s_j(x-x_j) \quad \text{avec} \quad s_j=\frac{y_{j+1}-y_j}{x_{j+1}-x_j}$$

figure 26 : **détail de la méthode de Stineman**



En fait, on fait passer une hyperbole d'équation (X-a)*(Y-b*X+c) = d, où a, b, c, d sont des paramètres calculés de telle sorte que l'hyperbole obtenue passe par les points I et J et aient comme tangentes en I et en J, les tangentes préalablement calculées.

- Si il y a un point d'inflexion, l'équation de la courbe est plus sophistiquée.

$$y=y_0+\frac{\Delta y_j \Delta y_{j+1}(x-x_{j+1}+x-x_j)}{(\Delta y_j-\Delta y_{j+1})(x_{j+1}-x_j)}$$

VI.A.3.2) AVANTAGES ET INCONVÉNIENTS DE LA MÉTHODE DE STINEMAN

avantages :
- la méthode est simple à mettre en oeuvre informatiquement.
- elle donne toujours des résultats cohérents, agréables à l'oeil, ce qui est la première qualité attendue.

inconvénients :
- cette méthode n'assure qu'une continuité d'ordre 1, ce qui numériquement peut être insuffisant, quand des dérivées d'ordre élevé sont demandées, et donne lieu parfois à des cassures de courbure.
- c'est plutôt une recette qu'une méthode, une étude systématique en est difficile, en particulier au niveau des erreurs.

VI.A.3.3) AUTRES MÉTHODES

Une étude a été publiée en 1993 (Hung 1993) pour dresser l'état de l'art des interpolations à base de cubiques (polynômes de degré 3), qui préservent la monotonie. En fait, on est aujourd'hui capable de définir des algorithmes basés sur des cubiques, qui permettent de faire des interpolations exactes jusqu'au 4ième ordre, c'est-à-dire que l'erreur commise sur les dérivées aux points de définition est d'ordre 3.

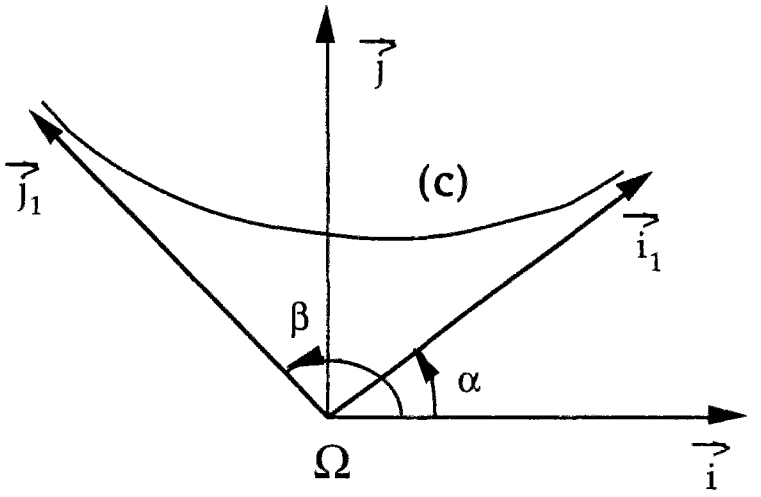
Cette étude systématique montre que les courbes obtenues ont des allures correctes, sans oscillations, et que les erreurs sont bornées et mathématiquement fondées. Ces courbes sont continues et dérivables, mais la continuité des dérivées d'ordre plus élevé n'est pas assurée.

Ces algorithmes n'ont pas été implantés dans PRINCE, mais ont un comportement apparemment au moins aussi bon que celui de Stineman.

VI.A.4) MONOTONIE ET CONCAVITÉ

Proposition : un changement de repère adéquat permet de transformer un arc de courbe qui a un sens de concavité donné en un arc de courbe de même sens de concavité et de plus monotone.

figure 27 : **transformation d'une courbe convexe en courbe convexe et décroissante**



Prenons l'exemple d'un arc de courbe convexe dans le repère (Ω, i, j), que nous allons rendre de plus décroissant dans le repère (Ω, i_1, j_1). Le changement de repère conduit à l'équation matricielle suivante :

(A) $$\begin{pmatrix} x_1 \\ y_1 \end{pmatrix} = \frac{1}{\sin(\alpha-\beta)} \begin{pmatrix} -\sin\beta & \cos\beta \\ \sin\alpha & -\cos\alpha \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

Supposons que la courbe (C) s'exprime par une fonction f de x dans le repère (Ω, i, j), et par une fonction g de x_1 dans le repère (Ω, i_1, j_1).

On montre sans difficulté, moyennant les bonnes hypothèses de dérivabilité, en repartant de l'équation (A), que :

$$\frac{dg(x_1)}{dx_1} = \frac{tg\,\alpha - \dfrac{df(x)}{dx}}{-tg\,\beta + \dfrac{df(x}{d(x)}} \cdot \frac{\cos\alpha}{\cos\beta}.$$

On choisit les axes de telle sorte que :

$$tg\beta < \frac{df(x)}{dx} < tg\alpha.$$

Dans ces conditions, la dérivée de g par rapport à x_1 ne s'annule pas dans l'intervalle de définition de g, ce qui prouve que g est une fonction strictement monotone sur cet intervalle.

CHAPITRE VII

CONCLUSIONS

VII.1) PRINCIPAUX RÉSULTATS DE CE TRAVAIL

Partant d'une application d'aide à la conception de réacteurs de raffinage, nous avons mis en place une architecture à base de tâches, réutilisable par d'autres, développé un outil de visualisation de courbes et d'analyse de points expérimentaux, qui intègre de nouvelles méthodes de tracé et approfondit la réflexion sur ce qu'est un système expert de conception.

VII.1.1) ARCHITECTURE À BASE DE TÂCHES

L'architecture que nous avons développée présente les aspects intéressants suivants :

• nous avons construit une **représentation 'objet' des raisonnements**, sous forme de **relation et de flux**.

Cette représentation s'appuie sur des définitions que nous avons voulues aussi rigoureuses que possible.

Elle est explicite et explicitée dans le document, pour qu'elle puisse servir de base à d'autres travaux et réflexions.

• L'architecture proposée est **indépendante du domaine** d'application, dans la mesure où la connaissance s'exprime sous forme de tâches. L'architecture peut facilement s'adapter à des objectifs variés, grâce à l'usage d'une méthode (EXECUTER) d'exécution des relations, et d'une méthode (VALEUR) de mise en correspondance des flux avec le domaine d'application. Ces deux méthodes peuvent être redéfinies sans problème, autant de fois que nécessaire.

• Les relations sont capables de s'**inverser** dès que c'est possible, ce qui permet de les utiliser dans n'importe quel sens. Cette faculté est importante pour les experts, qui peuvent ainsi se poser un problème direct ou inverse, comme ils en ont l'habitude dans leur pratique.

• L'architecture mise au point permet une **grande souplesse** d'utilisation et on l'emploie selon :

- un mode isolé, quand on exploite une seule source de connaissances,
- un mode enchaîné, quand on précise la suite des buts que le système doit atteindre,
- un mode automatique, quand on laisse le système rechercher une solution en chaînage arrière.

VII.1.2) COURBES

L'outil présent dans le système PRINCE est d'abord un outil de construction et de validation des connaissances. C'est en visualisant les courbes résultantes que les experts acceptent ou refusent les points expérimentaux.

• L'outil mis au point permet de visualiser et d'utiliser **n'importe quel type de courbes**, qu'elles se présentent sous forme de programmes, de formules ou de courbes définies par des points. L'outil est notamment connecté aux tâches utilisées dans les raisonnements par les experts, et ces deux aspects, tâches et courbes, se confortent et se complètent.

• **Deux méthodes nouvelles** d'affichage et de tracé de courbes ont été mises au point. Ces deux méthodes sont utiles quand on ne dispose pas d'équation. La première permet de tracer une courbe d'allure donnée dans un nuage de points. Par allure, il faut entendre concavité et monotonie. La seconde façon de faire exploite une base de cas, qu'elle interpole, en quelque sorte.

• L'environnement développé permet à l'utilisateur d'agir sur les courbes à partir de leur **définition**, en intervenant sur la valeur des points de définition par exemple, ou sur les formules, ou à partir de l'**affichage**, en modifiant directement l'allure du tracé. Cet environnement intègre également un outil d'optimisation interactif, qui permet d'afficher les résultats de l'optimisation, de les accepter ou de les refuser.

VII.1.3) INTELLIGENCE ARTIFICIELLE ET CONCEPTION

En nous demandant comment caractériser un système expert de conception, et comment PRINCE était lié à la conception, nous avons proposé une classification des systèmes informatiques et élaboré une démarche de résolution pour la conception.

• Notre **classification** s'appuie sur l'analyse des systèmes. Un système peut se voir comme une boîte noire qui transforme des entrées en sorties. Une application informatique, qui cherche à élaborer la boîte, se rangera plutôt dans la catégorie conception.

• Notre **proposition de démarche** part de l'idée que le problème initial se décompose en sous-problèmes qui font l'objet de méthodes spécifiques et qui peuvent donner lieu chacun à l'émission d'hypothèses.

VII.2) PROLONGEMENTS

VII.2.1) ARCHITECTURE À BASE DE TÂCHES

La principale limitation de l'architecture que nous proposons, est en fait, qu'elle n'a pas été testée sur d'autres exemples que le nôtre. Aussi le premier prolongement souhaitable est de la confronter à d'autres domaines, autres que le génie chimique.

La partie 'moteur', au moment de la résolution, n'a pas été non plus particulièrement travaillée. La raison en est que les tâches dans PRINCE, ne sont pas suffisamment nombreuses ou ne consomment pas un temps suffisamment long et que cet aspect n'est pas critique, pour le moment.

La partie contrôle, c'est-à-dire d'enchaînement des inférences, devrait permettre les enchaînements procéduraux classiques des connecteurs : et, ou, si ... alors, tant que. Ces enchaînements devraient pouvoir être définis également sur les buts, pour grader toute la souplesse de l'architecture proposée.

D'autres types de tâches pourraient être introduits dans l'application, qui gagnerait ainsi en généralité. Nous avons montré au chapitre V, que le plus difficile était d'écrire la méthode 'EXECUTER' correspondante.

Des outils pourraient être mis à la disposition des utilisateurs pour qu'ils puissent créer eux-mêmes les relations et les flux dont ils ont besoin, ceci étant encore à la charge d'informaticiens ou de personnes spécialisées.

VII.2.2) COURBES

Malgré tout le soin apporté à la réalisation, l'outil 'COURBES' manque encore de quelques fonctionnalités d'interface, qui rendraient son utilisation encore plus agréable.

En fait, cet outil constitue un ensemble homogène en soi, qui pourrait être détaché de l'application, et devenir un véritable progiciel indépendant.

Au-delà de cet aspect bien classique de réalisation, les méthodes de tracé proposées pourraient être largement affinées : en premier lieu, elles devraient être testées, encore une fois, sur d'autres exemples que ceux que nous avons proposés, pour comprendre leur comportement dans tous les détails. En second lieu, les variantes possibles de ces méthodes devraient être étudiées : il n'est pas invraisemblable que des changements minimes d'algorithme donnent lieu à des améliorations perceptibles et utiles.

VII.2.3) INTELLIGENCE ARTIFICIELLE ET CONCEPTION

La classification que nous proposons devrait, à notre avis, être confrontée à de nombreux autres exemples d'applications. Notamment, il serait utile de voir quelles sont les méthodes de résolution possibles pour chaque rubrique que nous proposons.

L'aspect temps est aussi à affiner. Il y a plusieurs façons de qualifier cet aspect, selon qu'on l'examine sous l'angle de l'historique, des tendances, des états, des réactions. Il faut se demander précisément en quoi le temps influe sur la classification présentée.

VII.2.4) CONCLUSION GÉNÉRALE

Clairement, le mémoire que nous présentons continue et prolonge une réflexion sur les différents aspects du métier de l'ingénieur. L'étude, la structuration, la représentation et la modélisation des tâches ont débuté depuis une dizaine d'années en Intelligence Artificielle, et se continuent aujourd'hui avec des tarvaux sur la coopération (Delouis 93)(Willamowski 94). Par ailleurs, des fonctions, aussi naturelles pour l'homme que de tracer des courbes, continuent encore aujourd'hui à susciter des recherches nombreuses. Ces deux aspects, tâches et courbes sont pourtant complémentaires, et indissociables dans la pratique quotidienne des ingénieurs.

RÉFÉRENCES BIBLIOGRAPHIQUES

(AFIA 94)
Dossier sur les applications de l'Intelligence Artificielle dans le pétrole et la chimie
Bulletin de l'AFIA n°19, novembre 1994

(Berlandier & 93) BERLANDIER P., TROUSSE B.
Expression et interprétation de métacontraintes pour le maintien de solution en conception
13ièmes Journées Internationales "Les Systèmes Experts et leurs Applications", Avignon 1993

(Bézier 87) BEZIER P.
Courbes et Surfaces
Mathématiques et CAO, Hermès, Paris, 1987, 2nde édition

(Boldo & 93) BOLDO F., BERNIS A., GONTHIER Y.
Apport d'un logiciel intégré comportant un générateur de système expert dans les problèmes de dimensionement
Récents progrès en génie des Procédés, Vol 7, 1993

(Boucot & 91) BOUCOT P., OLIVIER J.
Mise au point d'un système expert de conduite on line d'une unité de désalphaltage au propane, interfaçage avec un simulateur dynamique de procédé
Congrès MESUCORA, Paris, Novembre 1991

(Bourseau 93) BOURSEAU P.
Modélisation des connaissances de l'ingénieur de procédé : application de l'intelligence artificielle
Thèse de doctorat d'état de l'université Pierre et Marie Curie Paris 6, Janvier 1993

(Braunschweig 90) BRAUNSCHWEIG B.L.
Artificial Intelligence in the petroleum world
Revue de l'Institut Français du Pétrole, vol 45, n° 5, Septembre-Octobre 1990

(Brown & 89) BROWN D.C., CHANDRASEKARAN B.
Design Problem Solving : Knowledge Structure and Control Strategies.
London Pitman, 1989

(Cauvin & 93) CAUVIN S., BRAUNSCHWEIG B., GALTIER P., GLAIZE Y.
Model-based diagnosis for continuous process supervision : the ALEXIP experience
AI in Process Engineering 1993

(Cayeux 92) CAYEUX E.
Le système ODDA : intégration de programmation classique et d'intelligence artificielle
Revue de l'Institut Français du Pétrole vol 47 n° 3, mai-juin 1992

(Chaillot & 94) CHAILLOT M., CROCHON E.
Sélection Rationnelle de Méthodes dans un Environnement Réactif de Résolution de Problèmes
9ième congrès Reconnaissance des formes et Intelligence Artificielle, Paris, 11-14 Janvier 1994

(Chandrasekaran & 89) CHANDRASEKARAN B., JOSEPHON J., KEUNEKE A., HERMAN D.
Building Routine Planning Systems And Explaining Their Behaviour
International.Journal of Man-Machine Studies (1989)

(Chandrasekaran 90) CHANDRASEKARAN B.
Design Problem Solving, A Task Solving
AI magazine, winter 90

(Chandrasekaran & 92) CHANDRASEKARAN B, JOHNSON T.R., SMITH J.W.
Task Structure Analysis for Knowledge Modeling
Communications of the ACM, septembre 1992, pp124-137

(Clancey 85) CLANCEY W.J.
Heuristic Classification
Artificial Intelligence 27(3), 1985

(Clancey 92) CLANCEY W.J.
Model Construction Operators
Artificial Intelligence 53, number 1, January 1992

(Corby 91) CORBY O.
Le noyau multi-spécialistes NMS
Manuel de référence, INRIA, novembre 1991

(David & 90) DAVID J.M., KRIVINE J.P.
Structuration du raisonnement et explications
Revue d'Intelligence Artificielle vol n° 2 1990, pp77-88

(Dekker 92) DEKKER L.
Les tâches génériques selon Chandrasekaran
Université des sciences et techniques de Lille Publication ERA-92-116 1er Juin 92

(Delouis & 92) DELOUIS I., KRIVINE J.P.
Opérationnalisation du modèle conceptuel : vers une architecture permettant une meilleure coopération système-utilisateur
12ièmes Journées Internationales "Les Systèmes Experts et leurs Applications", Avignon 1992

(Delouis 93) DELOUIS I.
LISA : un langage réflexif pour la modélisation du contrôle dans les systèmes à base de connaissances. Application à la planification électrique.
Thèse de doctorat de l'université Paris 11, France, 1993

(Dhulesia 84) DHULESIA H.
Equation fits ASTM distillations,
Hydrocarbon Processing, pp 179-180, September 1984

(Dormoy 90) DORMOY J.L..
Méthodologies pour des systèmes experts de seconde génération
Architectures Avancées pour l'Intelligence Artificiell, onzièmes journées francophones sur l'informatique, 1990

(Garland 90) GARLAND W.M. J.
Knowledge-base design for heat exhanger selection
Engineering Applications of AI, 1990, Vol 3, September

(Goel & 89) GOEL A., CHANDRASEKARAN B.
Use of Device Models in Adaptation of Design Cases
Proceedings of a workshop on Case-Base Reasoning, Florida, May 31-June 2, 1989

(Gondran & 85) GONDRAN M., MINOUX M.
Graphes et algorithmes
édition Eyrolles, Paris, 1985

(Gondran 93) GONDRAN M.
Synthèse du numérique et du symbolique
Collection des notes internes de la direction des Etudes et Recherches, EDF, 93NJ00046, Mars 1993

(Hall 90) HALL M.R.
DESIGN : a generic configuration schell
The third international Conference on Industrial & Engineering Applications of Artificial Intelligence & Expert Systems, IEA/AIE 90

(Hartmann 93) HARTMANN F.
Process Control by an expert system at the Grand Puits refinery
13th International Joint Conference on Artificial Intelligence,Chambéry, France, 1993

(Hung 93) HUNG T. HUYNH
Accurate monotone cubis interpolation
SIAM Journal on Numerical analysis February 93 volume 30 number 1

(KADS 93)
KADS A principled approach to knowledge-based system development
Knowledge -based Systems, Vol 11, edited by Schreiber G., Wielinga B., Breuker J., Academic Press 1993

(Kelly & 93) KELLY B., CAILLOT P., ROEMER R., SIMIEN Thierry
SLURRY MINDER, A Rational Oil Completion Design Module
Innovative Applications of Artificial Intelligence 4, The AAAI Press, 1993

(Kolodner 93) KOLODNER J.
Case-Based Reasoning
Morgan Kaufmann Publishers, 1993

(Laidler 65) LAIDLER K.J.
Chemical kinetics
ed. Mac Graw Hill 2nde édition 1965

(Le Moigne 84) LE MOIGNE J.L.
La théorie du système général, théorie de la modélisation.
Paris, PUF 2nde édition 1984

(Lebart & 82) LEBART L., MORINEAU A., FÉNELON J.P.
Traitement des données statistiques, méthodes et programmes
Bordas, Paris, 1982

(Lesage 90) LESAGE G., CHACAR R.
SELIS : système expert pour la diffusion de la connaissance en technologie du génie chimique
10ièmes Journées Internationales "Les Systèmes Experts et leurs Applications", Avignon 1990

(Lindsay 93) LINDSAY R.K., BUCHANAN B.G., FEIGENBAUM E.A., LEDERBERG J.
DENDRAL : a case study of the first expert system for scientific hypothesis formation
Artificial Intelligence 61, 1993, 209-261

(Lunn & 91) LUNN K., ARCHIBAID I.G., REDFEARN J.J., ROBINSON A., BAMIGGBOYE A., COPE M.D., HIRD B.T.
An expert system for formulating lubrificating oils,
Artificial Intelligence in Engineering, 1991, Vol 6, N° 2, pp74-85

(MacDermott 82) MACDERMOTT J.
R1 : a Rule-based configurer of computer systems
Artificial Intelligence, 1982, 19(1), 39-88

(Maculet 91) MACULET R.
ARCHIPEL Intelligence Artificielle et Conception Assistée par Ordinateur en Architecture,
Thèse de doctorat de l'université Paris 6, spécialité informatique, 1991

(Miszessyn 90) MISZESSYN F.
Système Expert pour audit de four à ciment
Thèse de l'université Paris 6, spécialité génie chimique, 1990

(Mittal & 86a) MITTAL S., ARAYA A.
A Knowledge-Based Framework for Design.
In Proceedings of the 5th national conference on artificial intelligence, pp 856-865, Philadelphia, 1986,

(Mittal & 86b) MITTAL D., DYM C. L., MORJARIA M.
PRIDE: An expert system for the design of paper handling systems
IEEE Computer, vol 19, n° 7, p102-114, 1986

(Mittal & 90) MITTAL S., FALKENHAINER B.
Dynamic Constraint Satisfaction Problems
Proceedings Eigth National Conference on Artificial Intelligence, AAAI-90

(Montalban 87) MONTALBAN M.
Prise en compte de spécifications en ingénierie, application aux systèmes experts de conception
Thèse de doctorat de l'université de Nice, 1987

(Mostow 85) MOSTOW J.
Toward Better Models of the design Process,
AI Magazine 6, 1985

(Moulet 93) MOULET M.
ARC :Découverte empirique de lois numériques ou ABACUS Revu et Corrigé, Thèse de l'université Paris-Sud, 1993

(Myers & 88) MYERS D.R., DAVIS J.F., HERMAN D.J.
A Task-oriented Approach to Knowledge-based Systems for Process Engineering Design.
Computers and Chemical Engineering, vol 12, n° 9/10, pp 959-971, 1988

(Naraynan & 90) NARAYANAN N.H., GANDIKOTA M.S., MAROLDT J.
A process Design Framework based on task-structure analysis
Technical Report 1990, Laboratory for Artificial Intelligence Research, Department of Computer and Information Science, Ohio State University, Columbus, Ohio

(Neveu & 90) NEVEU B., TROUSSE B., COREY D.
SMECI: an Expert System Shell that Fits Engineering Design. Third International Symposium on Artificial Intelligence ITSM, oct. 90 Monterey Mexique

(Nii 86) NII P.
Blackboard Systems: The Blackboard Model of Problem Solving and the Evolution of Blackboard Architectures. (part one).
AI magazine 1986, vol 7, n° 2, pp 38-53

(Penalva 90) PENALVA J.M.
SAGACE : une représentation des connaissances pour la supervision des procédés continus
10ièmes journées internationales "les systèmes experts et leurs applications', Avignon, 1990.

(Penalva & 91) PENALVA J.M., COUDOUNEAU L., LEYVAL L., MONTMAIN J.,
DIAPASON : un système d'Aide à la Supervision
11ièmes Journées Internationales "Les Systèmes Experts et leurs Applications", Avignon 1991

(Pierret-Golbreich & 90) PIERRET-GOLBREICH C., DELOUIS I.
TASK : Task Architecture for the Structuration of Knowledge.
10ièmes journées internationales "les systèmes experts et leurs applications', Avignon, 1990.

(Plasson 91) PLASSON A., WAESELYNCK J.C.
Environnement de développement de tubes hyperfréquences
Revue technique Thomson CSF-vol 23-n° 4- Décembre 1991

(Press & 88) PRESS W.H., FLANNERY B.P., TEUKOLSKY S.A., VETTERLING W.T.
Numerical Recipes in C
Cambridge University Press 1988

(Pu & 91) PU P., RESCHBERGER M.
Assembly sequence planning using case-based reasoning,
Artificial Intelligence in Design 91, pp 171-187

(Raiman 92) RAIMAN O., SHIRLEY M.
Le Diagnostic à base de modèles
Cours n° 9, 12èmes Journées Internationales "Les Systèmes Experts et leurs Applications", Avignon 1992

(Riesbeck & 89) RIESBECK C.K., SCHANK R.C.
Case-Based Reasoning : An Overview,
Inside Case-based reasoning, Lawrence Erlbaum Associates Inc Publishers, Hillsdale, NJ, 1989

(Rousseau 88) ROUSSEAU B.,
Vers un environnement de résolution de problèmes en biométrie, Apport des techniques de l'intelligence Artificielle et de l'interaction graphique
Thèse de l'université Claude Bernard Lyon I, 1988

(Rousseau & 89) ROUSSEAU B., RECHENMAN F.
Apports des techniques de l'Intelligence Artificielle au développement d'environnements de résolution de problèmess
Actes de la Convention IA 89, Paris, 1989

(SADT 89)
SADT un langage pour communiquer
I.G.L. Technology, Eyrolles, Paris, 1989

(Schlaer & 88) SCHLAER S., MELLOR S.J.
Object Oriented Systems Analysis, Modeling the world in data
Yourdon Press 1988

(Siletti 90) SILETTI C.A.
Design of Protein Purification Processes by Heuristic Search
Aritificial Intelligence in Engineering 1990

(Simon 69) SIMMON H.A.
The Science of Artificial
MIT Press, Cambridge, Massachussets 1969

(Simmons & 87) SIMMONS R., DAVIS R.
Generate, Test and Debug : combining associationnal rules and causal models
7th International Joint Conference on Artificial Intelligence, Milan, Italie, 1987

(Smeci 92)
SMECI, Version 1.65, Manuel de Référence, Mai 1992,
ILOG S.A. 2 avenue Galliéni, B.P. 85, 94253 GENTILLY FRANCE

(Sriram & 89) SRIRAM D., STEPHANOPOULOS G., LOGCHER R., GOSSARD D., GROLEAU N., SERRANO D., NAVICHANDRA D.
Knowledge-based System Applications in Engineering Design : Research at MIT
AI Magazine, Fall 1989

(stage BRP 89)
BRUT-RAFFINAGE-PRODUITS manuel du STAGE
École Nationale Supérieure du Pétrole et des Moteurs / Formation Industrie, 1989

(Stephanopoulos 90) STEPHANOPOULOS G.
Artificial inteligence in process engineering - current state and future trends
Computer and Chemical Engineering, 14, pp 1259-1270, 1990

(Steel 90) STEELS L.
Components of Expertise.
AI magazine, 1990, vol 11, n° 2, p. 29-49.

(Stineman 80) STINEMAN R.W.
A Consistensly Well-Behaved Method of Interpolation,
Creative Computing, pp 54-57, July 1980

(Stolze 91) STOLZE M.
Task Level Frameworks for Cooperative Expert System Design
AICOM Vol 4 N°2/3 June/September 1991

(Subramanian 93) SUBRAMANIAN D.
Conceptual Design and Artificial Intelligence
13th International Joint Conference on Artificial Intelligence,Chambéry, France, 1993

(Thoraval & 90) THORAVAL M., MAZAS P., CHATENET J.P.
ARCHIX, système expert d'aide à la conception automobile
Actes de la Convention IA 90, Paris, 1990

(Trambouze & 84) TRAMBOUZE P., VAN LANDEGHEM H., WAUQUIER J.P.
LES RÉACTEURS CHIMIQUES conception/calcul/mise en oeuvre
Éditions technip 1984

(Trousse 89) TROUSSE B.
Coopération entre systèmes à base de connaissances et outils de CAO : l'environnement multi-agent ANAXAGORE
Thèse de doctorat, université de Nice Sophia-Antipolis, France, Dec 89

(Tsang 91a) TSANG J.P.
Automatisation de la conception par des techniques IA,
Cours n° 11, 11ièmes Journées Internationales "Les Systèmes Experts et leurs Applications", Avignon 1991

(Tsang 91b) TSANG J.P.
A combined generative and patching approach to automate design by assembly, Artificial Intelligence in Design'91, Edimburgh

(Vacher 92) VACHER P.H.
Cours de simulation en génie des procédés
École des Mines de Saint-Etienne, 1992-1993

(Vescovi & 90) VESCOVI M., TOMASENA M.
Générateur de systèmes experts d'aide à la conception caractérisés par une structuration hiérarchique des concepts
Actes de la Convention IA 90, Paris, 1990

(Visser 91) VISSER W.
The Cognitive Psychology Viewpoint on Design : examples from empirical studies
Artificial Intelligence in Design'91 Edimburgh, Scotland, 1991

(Vogel 88) VOGEL C.
Génie Cognitif.
Paris, Masson, 1988

(Wahl 92) WAHL F.
PRINCE, a knowledge-based system to assist process development
European Symposium on Computer Aided Process Engineering-3, 5-7 July 1993, Graz, Austria, pp 171-176

(Wahnschafft & 91) WAHNSCHAFFT O.M., JURAIN T.P., WESTERBERG A.W.
SPLIT : A Separation process designer
Computers and Chemical Engineering, Vol 15, N°8, pp 565-581, 1991

(Wahnschafft & 92) WAHNSCHAFFT O.M., LE RUDELIER J.P., BLANIA P., WESTERBERG A.W.
SPLIT : II . Automated synthesis of hybrid liquid separation systems
Computers and Chemical Engineering, Vol 16, Supplement, pp 305-312, 1992

(Waterman & 83) WATERMAN D., HAYES-ROTH F., LENAT D. (eds)
Building Expert Systems,
New-York, Addison-Wesley 1983

(Willamowski 94a) WILLAMOWSKI J.
Modélisation de tâches pour la résolution de problèmes en coopération système-utilisateur
9ième congrès Reconnaissance des formes et Intelligence Artificielle, Paris, 11-14 Janvier 1994

(Willamowski 94b) WILLAMOWSKI J.
Modélisation de tâches pour la résolution de problèmes en coopération système-utilisateur
Thèse de l'université Joseph Fourier Grenoble 1, spécialité informatique, Grenoble, avril 1994

(Winston & 81) WINSTON P.H., BERTHOLD K.P.H.,
LISP
Addison-Wesley Pubkishing Company, 1981

(Wuithier 72) WUITHIER P.
le pétrole, raffinage et génie chimique
Publications de l'institut français du pétrole, collections "sciences et techniques du pétrole" n° 5, tome 1, éditions technip, 1972 .

INSTITUT FRANÇAIS DU PÉTROLE

B.P. 311
92506 Rueil-Malmaison Cedex - France
Tél. : national (1) 47 49 02 14
international 33 (1) 47 49 02 14
Télex : IFP 634202 F
Télécopieur : 33 (1) 47 52 70 00

RÉSUMÉ

Dans le domaine du génie chimique, les ingénieurs tracent des courbes pour analyser les données recueillies. Une fois validée, cette connaissance est exploitée, en combinaison avec d'autres savoirs, sous forme de tâches. Cette thèse présente une architecture capable d'enchaîner n'importe quel type de tâches et de visualiser des courbes, appliquée à un problème d'aide à la conception de procédé de raffinage.

L'architecture proposée repose sur une analyse objets des raisonnements, où figurent les notions de relations (inversibles ou non) et de flux du point de vue statique, de problèmes et de tâches du point de vue dynamique.

Le module de visualisation exploite toutes les sortes de relations entre les variables et s'appuie sur des méthodes élaborées de tracé, dont deux sont nouvelles : la première s'inspire d'exemples a priori comme dans le raisonnement à base de cas, la seconde utilise les notions de monotonie et de concavité pour déduire des lignes dans un ensemble de points.

L'application est exposée dans le détail et conduit à une analyse des problèmes de conception, et nous avons développé notamment une nouvelle classification de ces systèmes.

Mots-clés : tâches, courbes, conception, procédés de raffinage

ABSTRACT

A help environment based on a task architecture and a curve visualization module. Application to the conception of refining processes.

In the field of chemical engineering, engineers trace curves to analyze experimental data. Once validated this knowledge is exploited, in combination with other information, in the form of tasks. This thesis presents an architecture capable of linking any kind of task and to draw the curves as a conceptual aid in the design of refining processes.

The proposed architecture is based on an 'object' analysis of the reasoning mechanism including the concepts of relations (inversible or not) and flux from a static point of view and the notions of problems and tasks from a dynamic point of view.

The visualization module employs all sorts of relations between the variables and relies on elaborate interpolation methods, of which two are new. The first is inspired by *a priori* examples as in case-based reasoning, and the second uses the notions of monotony and concavity to deduce lines within a group of points.

The application is exposed in detail and leads to an analysis of the problems of conception. In this sense we have developed a new classification of the conception systems.